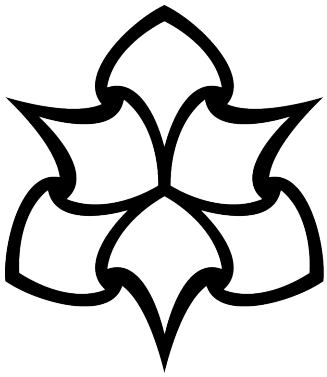

Manchester Metropolitan University

Eleyan, AA and Zhao, L (2011) Extending Web Service Architecture with a Quality Component: Web Service Architecture and Quality Component. LAP LAMBERT Academic Publishing. ISBN 3845418133

Downloaded from: https://e-space.mmu.ac.uk/618568/

Publisher: LAP LAMBERT Academic Publishing

Please cite the published version

https://e-space.mmu.ac.uk

Table of Contents

List of Figures

List of Tables

List of Code Listings

Abstract

The Web service technology provides standard mechanisms for describing the interface of the services available on the Web, as well as protocols for locating such services and invoking them. Each Web service has an associated Web Services Description Language (WSDL) document which describes how it works and how to invoke it. Such document is registered at a Universal Description, Discovery and Integration (UDDI) registry that provides a discovery service for the WSDL descriptions.

The Web services architecture consists of three components: Service Provider, Service Requester and UDDI Registry, and the interactions between them through *publish*, *find*, and *bind* operations. Between finding and binding steps there is another crucial step, which is not fully considered by current approaches. This is the step of *selection.* The UDDI service registry hosts hundreds of similar Web services, which makes it difficult for the service requesters to choose from them, as the selection is based on the functional properties only. However, many similar services are differentiated by their quality criteria. Therefore, quality criteria are important to be considered in the web service selection.

This thesis proposes a quality-based Web service architecture (QWSA) that extends the current Web service architecture with a quality server. The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The main purpose of quality server is to assist service requester to select the best available service that fulfils his/her preference by matching between a service requester's quality requirement and the service providers' quality specifications. In addition, this thesis reports the development of a quality matchmaking process (QMP) based on the proposed architecture by building a quality service selection system (QSSS). This QSSS has been verified and validated using a case study of Amazon E-commerce service (ECS).

Declaration

I herby declare that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning

Copyright Statement

(1) Copyright in text of this thesis rests with the Author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the Author and lodged in the John Rylands University Library of Manchester. Details may be obtained from the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such institutions may not be made without the permission (in writing) of the Author.

(2) The ownership of any intellectual property rights, which may be described in this thesis, is vested in The University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

(3) Further information on the conditions under which disclosures and exploitation may take place is available from the Head of School of Informatics.

ACKNOWLEDGMENT

I would like to express my gratitude to all those who made it possible for me to complete this thesis. Firstly and foremost, my heartily profound thanks gratitude and appreciation are addressed to my PhD advisor Dr. Liping Zhao for her encouragement, help and kind support. Her invaluable technical and editorial advice, suggestions, discussions and guidance were a real support to complete this thesis

Secondly, I would like to express my deepest gratitude and thanks to Dr. Ludmil Mikhailov for contributing in supporting this work with mathematical model, which without it this work will not be carried out. I would also like to thank AbdelBaki Djemi for offering his generous help and all of my friends in K13, K12, and K10 research laboratories for creating 'short breaks' during my research time in the university.

Thirdly special thanks to my husband Derar, who is also one of my lab-mates, and he has recently got PhD successfully. In addition, my son Loai and my beautiful daughters Dana, Alaa and my little baby Maryam whose patient and love enabled me to complete this work.

Finally yet importantly, I would like to give my best regards and love to my mother in-law, my sister in-law and my parents for their encouragement and help.

Chapter 1 Introduction

1.1 Motivation

The convergence of the World Wide Web (WWW) and the Extensible Markup Language (XML) [1] has increased the possibility for interoperable system-to-system communications and extended the role of the WWW from the information interaction to the service interaction. This convergence is leading to the development of the Web services technology.

The Web services technology enables software applications to communicate with each other in a platform and programming language in an independent manner over the Internet. Web services achieves system interoperability by exchanging an application development and service interactions using the XML–based standards such as Simple Object Access Protocol (SOAP) [2], Web Service Description Language (WSDL) [3] and Universal Description, Discovery and Integration (UDDI) [4].

As the popularity of Web services technology grows, the service requester is becoming increasingly aware of the importance of the service quality. Therefore, it is necessary for him/her to have a way of evaluating and selecting the services that meet his/her quality requirement. However, there are many challenges in establishing a quality-based service selection mechanism, including:

1. The service selection is still done by human clients, which is not desirable if thousands of services are available for selection [5].
2. The current service selection is only based on the functional information in the WSDL document. Service requester requires a selection mechanism that is based on functional information as well as non-functional information including the quality criteria such as availability, reliability, etc.

3. Quality criteria are dynamic in nature and depend on the characteristics of the providers' systems and the Internet.
4. Managing dynamic changes of quality criteria and ensuring up-to-date information.
5. Requester requires a mean to express his/her quality requirements and providers need a standard mean to express their quality specifications.

This thesis proposes a quality-based Web service architecture (QWSA) to address the above five challenges. This architecture incorporates a quality server that facilitates and assists service requester to discover and select the best published Web service. The quality server consists of four main components: quality manager, quality report analyzer, quality matchmaker and quality database.

The quality manager captures and manages the dynamic nature of the quality criteria to keep up-to-date information and save it in the quality database. The quality report analyzer produces statistical information about the service and store them in the quality database. The quality matchmaker is the core component that implements the quality matchmaking process (QMP) in order to match between the quality requirement that specified by service requester and the published quality specification of the services that specified by service providers to select the best service. The QMP is based on the mathematical model. A simulation programme called quality service selection system (QSSS) is developed to implement the QMP and to assist service requester to select the best service in an automated way.

Finally, this thesis has proposed a quality criteria classification that consists of four groups: Performance, Failure Probability, Trustworthiness and Cost. This thesis also has accommodated the quality classification within the Web Service Description Language (WSDL) to enable the service requester to express his/her quality requirements and the providers to express their quality specifications.

1.1.1 Research Scenarios

This section differentiates the notion of "Web services" and "service" in the coming two scenarios. The first scenario shows a selection of the best Web service based on the requester's quality requirements. A Web service in the first scenario has an interface that can be dynamically discovered using a service registry and can be invoked using SOAP messages protocol. After selecting and invoking the best Web service, the second scenario shows a selection of the best service provided by the previous selected Web service. A service in the second scenario could be service, product, Web site or any result.

The following two scenarios are used to motivate this thesis:

Scenario 1: Web service selection

The requester looks for a search engine Web services to search for books. There are four Web services as shown in Table 1-1: Amazon E-Commerce Web Services (ECS), Google Web Service, eBay Web Service and Yahoo Web service. The requester wants to select the best Web service with the following requirements:

- Throughput is the most important criteria.
- The requirement value of Throughput: High, Availability : High and Price : Low

Table 1-1 Web services

Quality Criteria	**Web Services**			
	Amazon	**Google**	**eBay**	**Yahoo**
Throughput/day	2200	1000	1440	1200
Availability	98	98	95	90
Price/month	0	0	5	0

After applying the mathematical model, which is described in Chapter 5, the weight of the quality criteria is:

$$W = [0.579 \quad 0.187 \quad 0.234]$$

It is noticed that Throughput criteria is the most important criteria which has the highest priority (0.579) then the Price (0.234) and the last is the Availability (0.187).

The output result that is based on the requester's quality requirements and preferences is shows in Table 1-2. It is seen that Amazon Web service (ECS) is the best one to select because its matching distance is the minimum "0.178". So ECS is the best Web service that the requester can select.

Table 1-2 Output of Web Service Selection

Web Services	Matching Distance
Amazon	**0.178**
eBay	**0.556**
Yahoo	**0.736**
Google	**0.96**

Scenario 2: Service selection

After selecting the Amazon E-Commerce Service (ECS) from scenario 1, the requester invokes it and uses it to select a service, where in this case is a book, regarding to its availability, seller reputation and its price. The requester wants to select the best book with the following requirements:

- The book's availability is the most important criteria from the requester's point-of-view.

- The requester wants a book with High availability, Medium seller reputation and Low book's price.

After applying the mathematical model, which is described in Chapter 5, the weight of the selected criteria is:

$$W = [0.723 \quad 0.07 \quad 0.206]$$

It is noticed that Availability criteria is the most important criteria which has the highest priority (0.723) then the Price (0.206) and the last is the Reputation (0.07).

Table 1-3 shows the ranking books from the least matching distance to the maximum. The matching distance is calculated using the mathematical model, which is described in Chapter 5. The service with the minimum distance is the best service to select. So, the book with the title "Service-Oriented Architecture" with matching distance "0.323" is the best book to select.

Table 1-3 Output of Book Selection

Product Name	Seller Name	Matching Distance
Service-Oriented Architecture	**hebertbooks**	**0.323**
Professional PHP Web Services	**hbytes**	**0.328**
Professional PHP Web Services	**westcoast_books**	**044**
How to Break Web Software	**studentbooks**	**0.52**

1.2 Research Questions

Web services technology offers many benefits; however, it creates significant challenges for application developers. One of the Web services challenges involves defining and guaranteeing the quality of the Web service. Before invoking a Web service, the service requester often wants to verify that the service will meet his/her expectations [6].

Unfortunately, current Web services technology is immature and still under development by the World Wide Web Consortium (W3C) and has the following challenges:

1. The current Web services environments do not offer comprehensive quality support as in the following:
    a. The UDDI is just a registry database and service discovery engine. It allows requester to look for Web services based on their functionality but not quality information.
    b. WSDL does not contain any information about quality criteria.
2. Selecting Web services over the Internet is difficult and challenging because it is not easy for the service requester to choose the best service of the same functional properties with different quality criteria information. Thus, effective automated technique for service matching and selection according to the service requester's quality requirement and preferences is needed.

Web services researchers are facing two research questions:

3. How to discover and select the desired Web services based on quality criteria?
4. How to specify the quality criteria using the Web services standards such as WSDL and UDDI?

1.3 Research Objectives

This thesis sets out to investigate the above two questions. The investigation will achieve the following seven objectives.

1. To create a quality criteria classification that organizes the most important quality criteria into four groups: Performance, Failure probability, Trustworthiness and Cost.
2. To extend the Web Services Description Language (WSDL) with the quality criteria classification.

3. To develop a quality-based Web services architecture (QWSA) that extends the current Web service architecture with quality server.
4. To develop a quality matchmaker component within the quality server in order to facilitate and assist the requester to select the best service based on his/her quality requirements.
5. To develop quality matchmaking process (QMP) based on the mathematical model.
6. To develop a simulation system called a quality service selection system (QSSS) that implements the quality matchmaking process (QMP). The QSSS is a graphical user interface (GUI) to enable the service requester to specify his/her quality preferences and requirements.
7. To demonstrate the effectiveness of the QSSS in selecting the best candidate service via simulation scenarios.

1.4 Research Methodology

1.4.1 Research Context and Assumptions

This thesis develops a quality matchmaking process that assists the service requester to select the best advertised service based on his/her quality preferences and requirements.

The tasks of this thesis, with respect to the research objectives will include the following:

1. Extending the current Web service architecture with quality server that called the quality-based Web service architecture (QWSA).
2. Developing a quality matchmaker component within the quality server.
3. Developing a simulation system called a quality service selection system (QSSS).
4. Using an Amazon E-Commerce Service (ECS) as a case study and applying it into the QSSS.
5. Evaluating the efficiency of the QSSS through simulation scenarios.

This thesis uses the following assumptions to demonstrate the new proposed concepts:

1. Only one requester at a time can query the QWSA architecture to select the best advertised service.
2. The values of the quality criteria are already measured or calculated when selecting the service.
3. The query which is sent by the service requester to QWSA architecture is volatile that is no new services will be added to UDDI and no changes to the quality criteria values for these services during the service selection process.

1.4.2 Concepts and Terminology

This thesis adopts the IBM Web services architecture to be extended with the quality server. The IBM Web services architecture is based upon the interactions between three roles: service provider, service requester and service registry. The interactions involve the publish, find and bind operations [7], [8]. Also, this thesis adopts the W3C Web services standards: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI). SOAP [2, 9] is an XML-based communication protocol for exchanging structured information in a decentralized, distributed system. WSDL [9] is an XML-based interface definition language for describing

the services (their interfaces) in a standardized manner. UDDI [10], [11], [12] is a Web services registry and discovery mechanism, which enables developers and businesses to publish and locate Web services on a network.

The IBM Web services architecture does not support the quality criteria. The UDDI service registry hosts hundreds of similar Web services, which makes it difficult for the service requesters to choose from them, as the selection is only based on the functional properties. The similar services are differentiated by their quality criteria. Quality criteria are important to be considered in the web service selection [13].

To address the above shortcomings, this thesis extends the IBM Web service architecture with quality server and calls it quality-based Web service architecture (QWSA). The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The main purposes of the QWSA architecture are to:

- Enhance the current UDDI role by enabling service publishing and discovering based on quality criteria.
- Match the quality specifications of the advertised Web services against the quality requirement that specified by the service requester.
- Assist the service requester to choose the best available service based on his/her quality requirements and preferences.

To achieve the above purposes, the following developments are required:

1. Construct a quality criteria classification that captures the most important quality criteria.
2. Extend the WSDL with quality criteria classification.
3. Develop a quality matchmaking process (QMP) that measures the distance between the quality requirements that specified by the service requester and

the quality specification that specified by the service providers and select the best match Web service with the minimum distance.

This thesis organizes the most important quality criteria into four groups under a classification called quality criteria classification. These four groups are: Performance, Failure probability, Trustworthiness and Cost. Each criteria group contains sub-criteria that hold the same characteristics. Performance criteria group contains the following sub-criteria: capacity, response time, throughput and execution time. Failure Probability criteria group contains the following sub-criteria: availability, reliability, accessibility and scalability. Trustworthiness criteria group contains the following sub-criteria: security and reputation. Cost criteria group contains the following sub-criteria: service price and execution price.

The quality criteria classification is implemented using XML Spy editor and the WSDL is extended with quality criteria classification by adding a new element <QualityCriteria> in its <service> element.

1.4.3 Theories used in this Thesis

This thesis develops a core component within the quality server which is called the quality matchmaker component. It contains the following sub-components: interface matchmaking, quality matchmaking and mathematical matchmaking. The quality matchmaker component matches the quality requirement of the service requester with the quality specification of the service providers in order to select the best match Web service. The quality matchmaker component performs the quality matchmaking process (QMP) to select the best service.

The QMP consists of four algorithms or filters: interface matchmaking, quality type matchmaking, quality value matchmaking and mathematical matchmaking algorithm. Each of these algorithms or filters narrows a set of matching candidates with respect to a given algorithm or filter criterion.

The mathematical matchmaking algorithm is the most important step that uses a mathematical model in order to select the best candidates Web service based on requester's quality requirements and preferences. Two techniques are used in the mathematical model:

1. Analytical Hierarchy Process (AHP) calculates the criteria weights based on requester's preferences.
2. Euclidean distance measures the distance between the requester's quality requirements and the providers' quality specifications. Web service with minimum distance is considered as the best service to select.

The QMP is implemented using Windows Application and C# language within Microsoft Visual Studio .NET 2003 software product to develop the quality service selection system (QSSS). The QSSS is a user interface that facilitates the service requester to specify his/her quality criteria preferences and requirements and display the best service to select.

This thesis uses Amazon E-Commerce Service (ECS) as a case study that is applied on the QSSS simulation system. The efficiency of QSSS is evaluated by comparing between selecting the best book from ECS without using QSSS and selecting the best book from ECS using QSSS. In addition, four scenarios are applied on the QSSS simulation system to evaluate the efficiency of the QSSS system.

1.5 Research Contribution

This thesis provides the following five contributions:

1. Definition of a classification of quality criteria

The most important quality criteria are organized in chapter 3 into four groups: Performance, Failure probability, Trustworthiness and Cost. Each criteria group contains sub-criteria quality that holds the same characteristics. Performance criteria group contains the following sub-criteria: capacity, response time,

throughput and execution time. Failure Probability criteria group contains of the following sub-criteria: availability, reliability, accessibility and scalability. Trustworthiness criteria group contains the following sub-criteria: security and reputation. Cost criteria group contains the following sub-criteria: service price and execution price.

The classification is generic that can be applicable in various domains and extensible, in which new criteria group and sub-criteria can be added without fundamentally altering the mathematical model and the service selection techniques that build on top of the classification.

2. **Extension of the WSDL with the quality criteria Classification.**

The above quality classification is implemented using XML Spy in order to design Quality Criteria XML Schema. The Quality Criteria XML Schema is augmented in the *Service Implementation Document* part of the WSDL by adding a new element <QualityCriteria> element in the <service> element. This extension enables the service requester to express his/her quality requirements when sending a request and the providers to express their quality specifications through publishing the services.

3. **Development of a quality-based web services architecture**

This thesis proposes a quality-based Web service architecture in chapter 4 that extends the current Web service architecture with quality server, because the current Web service architecture does not offer comprehensive quality of the Web service support. The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The quality server facilitates and assists service requester to discover and select the best published Web service.

4. **Development of a quality matchmaker component and quality matchmaking process**

The quality matchmaker component in the quality server is the core component in the proposed QWSA and it is well defined in Chapter 5. The quality matchmaker consists of the following three sub-components: Interface matchmaking, quality criteria matchmaking and mathematical matchmaking.

A quality matchmaking process (QMP) has been introduced in chapter 5 in order to select the best service. QMP consists of four algorithms: interface matchmaking, quality criteria matchmaking, quality value constraints matchmaking, and mathematical matchmaking algorithm.The mathematical matchmaking algorithm is the most important step that is based on the mathematical model. Two techniques are used in the mathematical model: Analytical Hierarchy Process (AHP) and Euclidean Distance.

QMP is implemented in Chapter 6 by building a simulation program called quality service selection system (QSSS). QSSS is developed by using C# Windows application in the Visual Studio .NET 2003 tool as a graphical user interface (GUI) to enable the service requester to specify his/her quality requirements.

5. **Publication**

This project has published the following paper:

- A. Eleyan, L. Mikhailov, and L. Zhao, "Quality-of-Service Support in Web services Architecture," *ISI*, vol. 9, 2004

1.6 Thesis Organization

The remaining thesis is presented in the following seven chapters.

Chapter 2: Background Studies

This chapter provides an overview of Web service architecture and its standards. It shows that Web services technology offers many benefits that provide more advantages over the distributed-computing technologies. For example, Web service is interoperable which has the ability to communicate and share data with software from different vendors and platforms. However, Web services

technology also has some challenges. The Web services standards are still immature and under development and do not offer quality criteria support. To address the Web service challenges, this thesis extends the current Web service architecture with quality server and develops a quality service selection approach.

Chapter 3: Quality Definition

This chapter introduces the definition of quality criteria in Web services syntax. It proposes a quality criteria classification that organizes the most important quality criteria into four groups. These groups are: Performance, Failure Probability, Trustworthiness, and Cost. The quality criteria classification is required in order to enable the service requester to specify his/her quality requirements. Also, it extends the current Web Service Description language (WSDL) with the quality criteria classification by adding a new element tag called <QualityCriteria>.

Chapter 4: QWSA: A Proposed Quality-Based Web Service Architecture

This chapter provides an introduction to a quality-based Web service architecture. (QWSA) which extends the current Web services architecture with quality server. The quality server acts on behalf of the requester to select the desired Web services. It consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The role of each component is elaborated.

Chapter 5: A Theoretical Model of Service Selection

This chapter introduces the core component in the quality server of the proposed quality-based Web service architecture (QWSA), which is the quality matchmaker.

The quality matchmaking challenges in UDDI and Web service environment are introduced. The quality matchmaker sub-components and its roles are described. The quality matchmaking process (QMP) is developed based on the mathematical model. The mathematical model uses two techniques: Analytical Hierarchy

Process (AHP) and Euclidean distance in order to select the best candidate Web services based on requester's quality preferences and requirements.

Chapter 6: Implementation of the Quality Matchmaking Process

This chapter presents an implementation of the quality matchmaking process (QMP), which is applied by the quality Matchmaker component.

The QMP is implemented by using Windows Application and C# language within Microsoft Visual Studio .NET 2003 software product. to develop the quality service selection system (QSSS). The QSSS is a user interface that facilitates the service requester to specify his/her quality criteria preferences and requirements and to display the best service to select.

Chapter 7: Evaluation

This chapter evaluates the proposed QWSA architecture, the QMP process and the QSSS simulation system.

The QWSA is evaluated by comparing it with the related architectures. In the related architecture, the QoS brokers are introduced between the service requester and the service providers. The QoS brokers are not well defined; they do not describe the details of the service selection process.

The QMP is evaluated by comparing it with the related approaches. It is seen that most of the related quality service selection approaches varies in the previous work from semantics approaches to computation approaches. The proposed quality matchmaking process (QMP) in this project is based on the mathematical and it considers the service requester's quality preferences and requirements.

The QSSS is evaluated through applying Amazon E-Commerce Sevice (ECS) as a case study.

Chapter 8: Conclusion and Future Work

Chapter 1 Introduction

This chapter shows that the contributions that has been achieved in this project. Further investigation needed on some aspects which is out of the scope of this thesis.

Chapter 2 Background Studies

2.1 Introduction

This chapter provides an overview about web services architecture and their standards. Section 2.2 explains the differences between Web services and the traditional distributed computing components. Web services architectures provide a framework for developing, and deploying loosely coupled applications. It enables opening business-to-business (B2B) and application-to-application interactions on the Web, based on existing Web protocols and on open XML standards. Section 2.3 introduces an overview of Web services architecture and their technologies. Section 2.4 introduces the tools used to implement Web service technology, as Microsoft's .NET and Sun Microsystems' J2EE (Java 2 Platform, Enterprise Edition). A comparison of these tools is presented. Section 2.5 introduces the Web services challenges. Section 2.6 introduces the semantic web and web services. Section 2.7 discusses the related work which illustrates different approaches that support quality issues in Web services technology.

2.2 Web Service History and Evolution

The combination of conventional middleware technologies such as OMG CORBA, Microsoft COM+ or Enterprise JavaBeans and Web technologies supports the integration of business processes and applications. This combination has become insufficient because it does not consider an integration of different data models or business rules. Enterprise Application Integration (EAI) has tried to solve the aforementioned issue and has become widely spread in B2B environments [14]. However, the EAI solutions are complex to use, and do not provide interoperable solutions. For example, it is impossible to invoke a CORBA [15] servant from a Web-based COM client. Therefore, there is a need to find an alternative solution to the application integration with simplicity and

interoperability. This solution is to build Broker-based middleware using Internet protocols such as Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML). This is the essence of Web services [16], [17]. Hence, Web services technology is considered as the traditional distributed architecture by addressing the issue of limited interoperability.

The initial ideas for Web services had been started by IBM and Microsoft. In 1990s, with the development of the World Wide Web (WWW), the information technology (IT) and communications industry can work together using a common framework including Transmission Control Protocol (TCP) and HTTP protocols. However, the creation of XML has paved the way to Web services [18], [19].

XML was developed by an XML Working group originally known as the Standard Generalized Markup Language (SGML) in 1996. In 1998, the XML version 1.0 specification was accepted as a World Wide Web Consortium (W3C) Recommendation, which means that the technology is stable for deployment in industry [1], [6]. XML documents contain data, but no formatting instructions, so applications that process XML documents must decide how to display the document's data. Software developers are integrating XML into their applications to improve Web functionality and interoperability [6].

The next stage was the development of the simple object access protocol (SOAP)- the standardized message-passing protocol based on XML- by Microsoft. SOAP was conceptualized in 1998 and published as SOAP 0.9 in 1999. The newest version of SOAP is SOAP 1.2 which is currently being defined by the W3C. The purpose of SOAP is to enable data transfer between peers in a decentralized, distributed environment using XML [6].

Software vendors realized that applications calling services across a network need information about a specific service before interacting with it. Therefore, in March 2001, Microsoft, IBM and Ariba submitted Web Services Description Language (WSDL) 1.1 to the W3C. Nearly every Web services published on the Internet is

accompanied by an associated WSDL document, which defines the kinds of messages a Web service can send and receive [3], [6].

With SOAP and WSDL, companies can create and describe their Web services. In March 2000, IBM, Microsoft, and Ariba started working on tools for discovering available Web services, and in September 2000, the first version 1.0 of the Universal Description, Discovery, and Integration (UDDI) was published. UDDI version 2.0 was released in June 2001. The UDDI version 3.0 was published in July 2002 [4]. UDDI simplifies the process of creating B2B relationships and connecting electronic systems to exchange data and services[6].

The Web services with its core technologies SOAP, WSDL, and UDDI, provide a language-neutral, environment-neutral programming model that accelerates application integration inside and outside the enterprise [20]. By the end of 2000, the major IT software infrastructure vendors announced their commitment to web services. Oracle, HP, Sun, IBM, BEA, and Microsoft support and deploy the Web services standards in their products [18].

2.2.1 Service-Oriented Architecture (SOA) and Web Services

Service-Oriented Architecture (SOA) is an approach that represents application or software functionality as services on the network [21]. These services can communicate with each other either for passing data or coordinate some activity inside or outside organizational boundaries [22], [23]. The SOA based on Web services has solved the limitation of the distributed computing technologies such as Common Object Request Broker Architecture (CORBA) and the Distributed Component Object Model (DCOM) in that they are tightly coupled, which means any change to one tightly coupled system always affects the whole architecture. Whereas, the Web services are loosely coupled, which means the developer can make changes to a Web services without impacting the whole architecture. The service requester binds the service provider in a loosely coupled manner this means the service requester has no knowledge of the provider's programming

language or deployment platform. The service requester invokes services by using messages (request and response messages) rather than using Application Programming Interface (APIs) [23].

Web services technology is considered as the convergence between the service-oriented architecture (SOA) and the Web. The Web services architecture takes all the best features of the service-oriented architecture and combines them with the Web [24]. Table 2-1 gives an overview of some important differences between Web services and the traditional distributed systems technologies. It shows that Web services technology supports universal communication using loosely coupled connections. Web services protocols are completely vendor, platform, and language-independent. Hence, the Web services architecture eliminates the constraints of DCOM, CORBA, and RMI, and supports Web-based access easy integration and service reusability [22]. Figure 2-1 shows the relation between the distributed computing technologies: CORBA, DCOM and RMI and the SOA that is expanded to include the Web services [25].

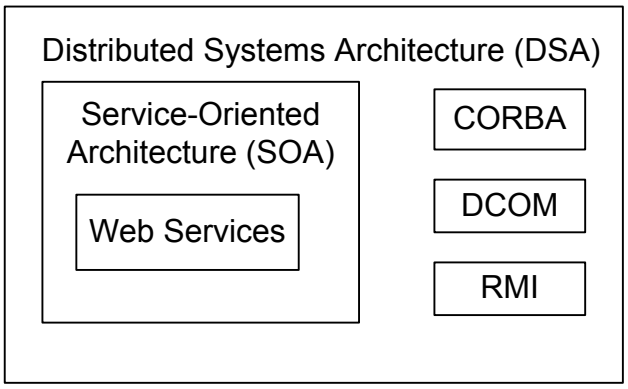

Figure 2-1 Service-Oriented Architecture Technologies

Table 2-1 Differences between Web Services and Distributed Systems

Web Services	Traditional Distributed Systems
Loose	Tight
Text Message	Binary
Vendor, platform and language independent	Vendor, platform and language dependent
interoperable	Limit in the interoperability
Flexible and reusable	Limit the flexibility and reusability

2.2.2 Web Services Definition

Web services definitions range from the very generic to very specific and restrictive. The generic Web services definition is an application accessible to other applications over the Web. This is an open definition which means anything has a URL is a Web service.

The more specific definition of Web services is the one provided by the World Wide Web Consortium (W3C) as "*a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artefacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols*" ([26] cited [27]).This definition shows that the Web services can be "defined, described, and discovered", which clarifies the meaning of "accessible". This definition also shows that Web services are components that can be integrated into more complex distributed applications using XML as a data format for Web-based interactions [26] .

Another more specific definition in the online technical dictionary Webopedia, as " *a standardized way of integrating Web-based applications using the XML,*

SOAP, WSD and UDDI open standards over an Internet Protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available" ([26] cited [28]). Specific standards (SOAP, WSDL, and UDDI) are mentioned here and used for binding and interacting with a Web services.

This thesis defines Web services as software components that use XML to exchange information and services (functionality) with other software via common Internet protocols (e.g., HTTP) over a network.

Web services technology is programmable, that encapsulates a task when an application passes data or instruction to it. Web services is based on XML which enables it to communicate with other applications, even if these applications are written in different programming languages and run on different platforms. XML bridging the differences between systems that use different component models, operating systems, and programming languages [6] .

A Web services exposes the following characteristics:

- A convergence of software (the World Wide Web) and network (the Internet) technologies [18].
- Accessible over the Internet.
- Can be invoked by another program using an interface.
- Can be registered and discovered via a Web service registry.
- Communicates using messages protocols.
- Supports loosely coupled connections between systems [29].

The Web services provides the following advantages [25]:

- It provides interoperability between various software applications running on different platforms. Therefore, the developers do not need to change their development environments in order to produce or consume Web Services.
- It uses open standards and protocols.

- It allows software and services from different companies and locations to be combined easily to provide an integrated service.
- It allows reusing of services and components within an infrastructure.
- It is loosely coupled that facilitating application integration.

2.2.3 Service Definition

The "service" in the context of "Web services" represents the function or the behaviour that is provided by a reusable software component in a business process [23], [30], [31]. A service has an interface and can be called from another program or service. It can be dynamically discovered using a service registry and can be invoked using SOAP messages protocol [29]. A service stresses interoperability and location transparency. Hence, the service implementation is hidden from the user and may be executed either on different computers in one enterprise or on different computers for a number of business partners [23], [31].

A Web service is a particular capability to communicate with other parties by transmitting and receiving information in a way that is fully specified with respect to: the requester's requirement, how the information is formatted (messages) and transmitted (using HTTP as a transfer protocol), how end-to-end exchanges of the information are effected.

The service description is divided into different levels:

- **Functional description**. The behaviour of the service in functional terms providing technical oriented description of the service.
- **Binding/interface description**. The definition of the service interface, the communication protocol needed to interact with it and the address associated with the protocols. This level of description is addressed by tModels, Binding Templates of Universal Description Discovery and Integration (UDDI) and Web Service Description Language (WSDL).

- **Transaction description**. A description of the service from a business point-of-view associated with quality issues, behaviour guarantees and usage of the service [32].

There are two types of Web services: services that support business-to-business interaction and services support business-to-customer interaction [33]. Some of the business-to-customer services are simple which returns simple result, e.g. a currency converter or a weather forecast, and complex services, e.g. flight booking or restaurant reservation. The simple service just converts for example US Dollars to Japanese Yen, whereas the a flight booking services, the result depends on the user needs [33].

2.3 Web Service Architecture

Web services architecture explores the principles behind the next generation of e-business architectures, presenting a logical evolution from object-oriented systems to systems of services [34]. Some of the fundamental concepts in Web services are as in object-oriented systems like encapsulation, message passing, dynamic binding, service description and querying.

There are many proposals and frameworks for Web services. The main three frameworks are IBM Web Services [7], Microsoft's .NET [35] and Sun Open Net Environment (ONE) [36]. Although, each of theses frameworks has its own particular position, they all share a common set of technologies such as SOAP, WSDL and UDDI. However, the IBM Web Services architecture is widely used in the industry [23, 29], and it will be described in the coming section.

2.3.1 IBM Web Service Architecture

IBM proposed a conceptual architecture for implementing Web services in terms of a service-oriented architecture [23]. The IBM Web services architecture is based upon the interactions between three roles: service provider, service requester and service registry. The interactions involve the publish, find and bind

operations [7], [8]. Figure 2-2 shows the interaction between service providers, service requester and service register in the publish, discovery, and consumption of Web services [37]. This architecture is described below.

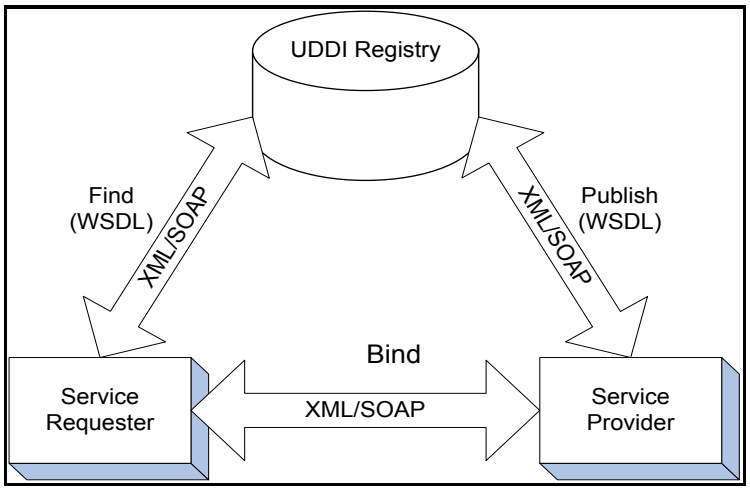

Figure 2-2 Web Service Architecture

Web services architecture roles are described below.

- **Service provider**. It is the owner of the Web services. It either represents the services of a business entity or represents the service interface for a reusable subsystem. The service provider defines a service description for the Web service and publishes it to a service register [29].
- **Service requester**. It represents a business application component that is looking for invoking or initiating an interaction with a service. The service requester uses a find operation to retrieve the service description from the service registry and uses the information in the service description to bind with the service provider and invoke the Web service implementation [7], [29].
- **Service registry**. It acts as a repository where service providers publish their service descriptions. Service requester find services and obtain static binding during development or dynamic binding during execution. For the static binding, the service registry is an optional role in the architecture. The service provider can send directly the service description to service requester, and the service requester can obtain a service description from other resources such as

FTP site, Advertisement and Discovery of Services (ADS) [7]. The Web services architecture operations are described below.

- **Publish**. A service description needs to be published so that the service requester can find it. The published location can vary depending upon the requirements of the application [7].
- **Find**. The service requester retrieves a service description directly or queries the service registry for the type of service required. The find operation can be involved in two phases: at design time to retrieve the service's interface description for program development, and at runtime to retrieve the service's binding and location information for invocation [7].
- **Bind**. The service requester invokes or initiates an interaction with the service at runtime using binding information in the service description to locate and invoke the service.

However, between finding and binding there is another essential operation, the current approaches ignore. This is the operation of service *selection* [38]. The UDDI service registry contains hundreds of similar Web services, which makes it difficult for the service requesters to choose from them, as the selection is only based on the functional properties. The similar services are differentiated by their quality criteria. So, quality criteria are important to be considered in the service selection [13]. The service selection operation is described below.

Service selection. It is the phase where a requester selects a service instance (implementing a discovered interface). Selection is based on non-functional attributes such as quality criteria. The quality criteria of a service (e.g. cost, response time) should be taken into account when selecting web services. This facilitates differentiation among services with the same functional characteristics and also gives some degree of confidence to the Web services' requestors about the quality of the service they are going to invoke. A service instance may be replaced by another at runtime if it doesn't meet the requester's needs.

The Web Services Stack

The Web services stack describes the relation between the Web service standards to their features (publish, find, and bind). Web services stack is built from layers of technologies and standards on which services can be implemented and deployed [29]. The upper layer is based on the layers below it. Figure 2-3 illustrates a Web services stack. It shows that this stack is a collection of standardized technologies (the text on the left) and application programming interface (APIs) that enable customers and applications to locate and utilize Web services. Figure 2-3 also illustrates how each layer facilitates the use of Web services [20].

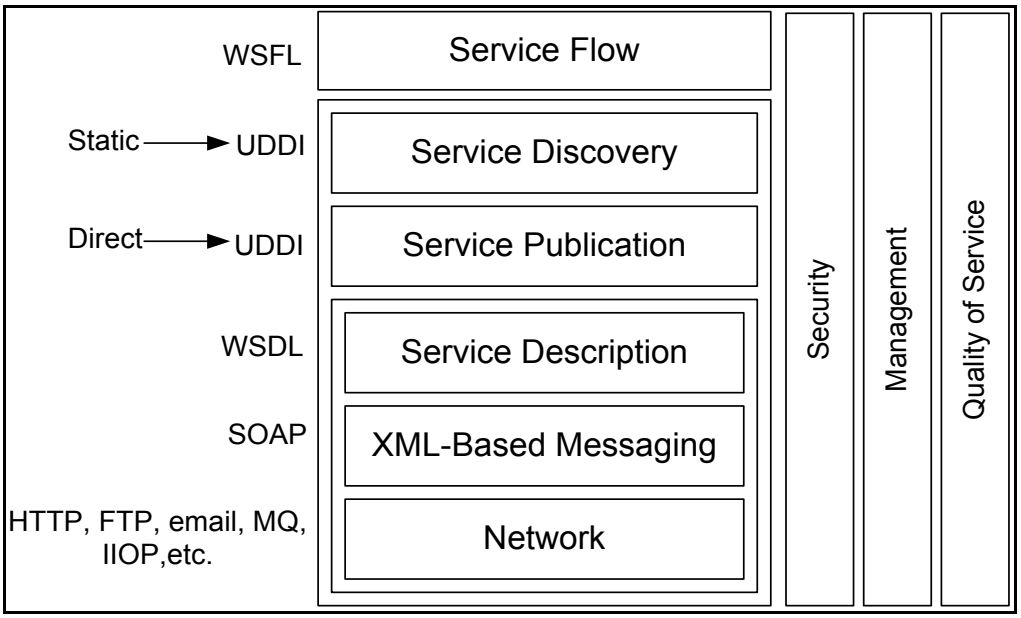

Figure 2-3 Web Services Stack taken from [7]

The network is the foundation layer for the Web services stack. Web services must be available and accessible over a network and use deployed network protocols such as HTTP and other Internet protocols like the Internet Inter-ORB Protocol (IIOP), SMTP, FTP, Message Queuing (MQ), and so on [7].

The next layer of that stack is an XML-based messaging layer that facilitates the communications between Web services and their clients [20]. The messaging layer is based on an XML messaging protocol SOAP [39]. SOAP messaging

protocol supports publish, find and bind operations in the Web services architecture [7].

WSDL [39] is an XML-based service description that describes available Web services to clients. These descriptions take the form of XML documents for the programming interface and location of Web services. WSDL defines the interface and mechanism of service interaction.

WSDL is a standard service description to support interoperable Web services. Additional description needed for example to specify the business context, quality of service and service-to-service relationship can be achieved by complementing the WSDL document with other service description documents. For example, business context can be described by using UDDI [39] data structure in addition to the WSDL document. Service composition and flow are described in a Web Services Flow Language (WSFL) document [7].

Because a Web service must be a network accessible via SOAP and represented by a service description, the first three layers represent the interoperable base stack that all inter-enterprise or public Web services should support. The remaining layers in the stack are optional and can be used as business needs require them [20].

Service publication is any action that makes a WSDL document available to a service requester. There are two Publishing mechanisms; direct publish and dynamic publish. In direct publish, the service provider sends the service description directly to the service requester, for example using email. In dynamic publish; the service provider can publish the WSDL document to a local WSDL registry, private UDDI registry or the UDDI operator node. The Web services descriptions can be retrieved from a given URL (pointer to the WSDL) [7].

Likewise, service discovery is any action that enables the service requester to acquire access to the service description and an associated functional description of the service and makes them available to the application at run time[40]. Acquiring Web service descriptions depends on how the service description is

published (direct publish or dynamic publish). Service requester can find the Web services during two phases of an application lifecycle- design time and runtime. At design time, service requester searches the type of interface that the Web service descriptions support. At runtime, service requester searches for a Web services based on how they communicate or the quality of advertised services.

Service requester can retrieve a service description from a service description repository, a simple service registry or a UDDI operator node at both design time and runtime. The look-up mechanism provides find operation by type of interface (based on a WSDL template), the binding information (that is, protocols), properties (such as QoS parameters), the taxonomy of the service, business information, and so on [7].

The topmost layer, service flow, describes service-to-service communications, transactions, and flows. IBM proposed a Web Services Flow Language (WSFL) to describe these interactions [41].

The vertical layers represent security, management, and quality of service, are supplied to meet the stringent demands of today's e-businesses. These vertical layers must be addressed at each layer of the stack.

The lower layers of the stack are relatively mature and more standardized than the higher layers. The Web services maturation and adoption will drive the development and standardization of the higher and the vertical layers of the stack [7].

2.3.2 Web Services Technologies

A web services relies on several enabling technologies including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and integration (UDDI). These technologies form the core of Web services technologies [26], [20] and accepted as the foundation for an open Web service framework.

Extensible Markup Language (XML)

XML is the basic foundation of Web services. It is a standard for describing data structure and formats (providing common syntax) [42], [43]. XML was defined by the W3C as an open standard technology [1].

XML is a "meta-language", that is a language for describing languages, that enables to design their own customized markup languages for different types of documents. Because XML is just text, any application can understand it as long as the application understands the character encoding in use. This makes XML a good choice for describing method invocations in a platform and language-neutral fashion [44], [42].

XML-based Web services communicate by using standard Web protocols like Simple Object Access protocol (SOAP) [2], Universal Description, Discovery and Integration (UDDI) [11], and Web Services Description Language (WSDL) [3] to define the interaction. These standards use XML interfaces and messages that enable any application to interpret. XML allows developers to create their own tags, enabling the definition, transmission, validation and interpretation of data between applications [45].

Simple Object Access Protocol (SOAP)

SOAP [2, 9] is an XML-based communication protocol for exchanging structured information in a decentralized, distributed system [46]. When an application interacts with a Web service, the interaction relies on messages as the basic unit of communication through which the two systems exchange data [6]. SOAP can turn a service invocation into an XML message, to invoke object methods provided by the service. The service then uses the information in the XML message to perform its function, and the Web service can return the result via another XML message [6]. The main goal of SOAP is to facilitate interoperability. Hence it is widely viewed as the backbone to a new generation of cross-platform; cross-language distributed computing architecture of Web services.

SOAP has the following characteristics:

- It is designed to be simple and extensible.
- It facilitates interoperable communication among computing systems in a decentralized, distributed network [6].
- It provides a framework to describe message content and process instructions, and an optional set of encoding rules for representing defined data-types.
- All SOAP messages are encoded using XML.
- It is transport protocol independent. Sine Hypertext Transfer Protocol (HTTP) is one of the supported transports. SOAP can be run over an existing Internet infrastructure.
- It is operating system independent and not tied to any programming language or component technology. It is object model neutral [25].

SOAP combines the data capabilities of XML with the transport capability of HTTP and supports a loosely coupled distributed data exchange.

SOAP is different from traditional distributed protocols in that the traditional distributed protocols such as IIOP, ORPC and JRMP are binary protocols whereas SOAP is a text-based protocol which makes it easier to debug and to read the binary stream.

SOAP Architecture

SOAP consists of the following four components:

- **The SOAP envelope**. It describes the format of a SOAP message.
- **Data Encoding Rules**. These rules encode data types of the data structures sent in a message. They enable applications that receive SOAP messages to recognize its format and therefore process it.
- **Remote Procedure Call Protocol**. This defines how a message can execute remote procedure calls (i.e., the requests to execute a program component on a

remote computer). Remote Procedure Call (RPC) is a technology that application can invoke (execute) the procedure (a set of instructions or methods) placing on another computer. SOAP also supports document-style communication in which no methods is invoked, it is used for notification and not required a response.

- **Binding Framework**. This defines the protocol through which SOAP message are transmitted to applications. HTTP is a common protocol used to transmit data over the Internet. Also, SOAP can use other protocols such as HyperText Transfer Protocol Secure (HTTPS) and Simple Mail Transfer Protocol (SMTP) [47], [6].

SOAP Message Structure

SOAP encapsulates data in messages that are transferred to and from Web services. Figure 2-4 illustrates the structure of a SOAP message, consisting of three parts [44]:

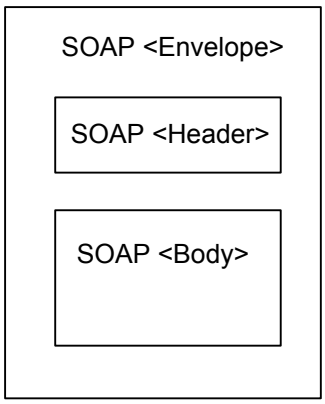

Figure 2-4 SOAP Message Structure

- **SOAP Envelope**. It is the outermost element in a message. It is the root of the XML document that defines a SOAP message.
- **SOAP Header**. It is a child element of the envelope. It may include additional features and functionality, such as security and quality criteria.

- **SOAP Body**. It is a child element of the envelope. It includes the actual data or instructions for tasks that receiver must perform, such as calling method or include information that must be processed by an application.

SOAP isn't the only way a requester can query database registry. The other method is known as REST, which stands for Representational State Transfer. REST is described below.

Making REST Requests

REST is an architectural style that was created by Roy Fielding in his Ph.D. thesis ([48] cited [49]). REST is not standard, but it uses standards such as HTTP, URL and XML. REST unlike SOAP, doesn't require installing a separate toolkit to send and receive data. Instead, the idea is to know to look for available Web services.

Amazon E-Commerce Service (ECS) is a case study (see Chapter 7 for details) in this thesis has both SOAP and REST APIs. It allows requesters to make calls to ECS by passing parameter keys and values in a URL (Uniform Resource Locator). ECS returns its response in XML (Extensible Markup Language) format. The developer can enter the REST URL into the browser's address bar, and the browser displays the raw XML response.

Web Services Description Language (WSDL)

WSDL [9] is an XML-based interface definition language for describing the services (their interfaces) in a standardized manner [26], [3]. The Web service published on the Internet is associated with WSDL document, which defines its location on the Web, data and message types, interaction patterns, and protocol mappings [8]. WSDL consists of two parts as shown in Figure 2-5: *service interface definition* and *service implementation definition*. The *service interface definition* is an abstract definition of a Web service, used to describe a specific type of service. The *service implementation definition* is a description of an actual service that implements the *service interface definition.*

WSDL Document Syntax

Each WSDL document contains XML elements that define the characteristics and capabilities of a Web service. These elements belong to one of two categories:

- Abstract definition, which define general concepts of the service that can be apply to more than one instance WSDL file. Abstract elements are related to Service Interface Definition (see Figure 2-5).
- Concrete definitions, which define specific concepts that apply to real interactions. Concrete elements are related to Service Implementation Definition (see Figure 2-5) [3], [50].

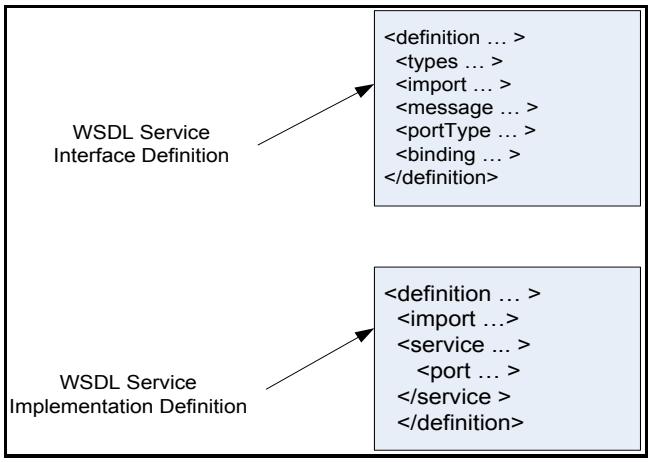

Figure 2-5 Components of a Service Description

The abstract definition is separated from concrete definitions which can be reusable. Figure 2-6 shows the two categories of the WSDL Main elements with their description [3], [50].

WSDL element	Element description
Abstract Definitions	
message	Provides a definition of the message that is communicated
portType	Defines the service interface of the operations that the Web service support.
operation	Describes an action providedby the Web service. Is a child of **portType**.
type	Provides definitions for the data types that SOAP messages contain.
Concrete Definitions	
Binding	Specifies the protocols by which nodes transport messages and for data encoding.
Port	Specifies the address for a particular **binding**. Is a child element of **service**.
Service	Specifies the actual location (URL) of the Web service on the server.

Figure 2-6 WSDL Main Elements

Hence, a WSDL document uses the following elements [3], [50] in the definition of network services:

- **Type**. It is a container for data type definitions using some type system (such as XSD).
- **Message**. It is an abstract definition of the data being communicated.
- **Operation**. It is an abstract description of an action supported by the service.
- **Port Type**. It is an abstract set of operations supported by one or more endpoints.
- **Binding**. It is a concrete protocol and data format specification for a particular port type.
- **Port**. It is a single endpoint defined as a combination of a binding and a network address.
- **Service**. It is a collection of related endpoints.

Universal Description, Discovery and Integration (UDDI)

UDDI is a Web services registry and discovery mechanism, which enables developers and businesses to publish and locate Web services on a network. It defines an electronic business registry where businesses can describe their business and register their Web services as well as discover and integrate with other businesses that offer Web services. UDDI is based on XML and SOAP. Interaction with UDDI is accomplished via SOAP interfaces. [10], [11], [12].

UDDI Architecture

- **UDDI Business Registry**. A core component of UDDI that implements the UDDI data model and API.
- **UDDI data model.** An XML schema for describing businesses and Web services.
- **UDDI API**. A SOAP based API for searching and publishing businesses and Web services.

The following is a detailed description of the UDDI Architecture**.**

UDDI Business Registry (UBR)

The UDDI Business Registry (UBR) consists of three components as shown in Figure 2-7:

White pages - Contains general information such as name, address and contact information.

Yellow pages - Contains industrial categorizations based on their products and services. For example, Software Company might be categorized under computer software or software engineering.

Green pages - Contains technical information about services and how to invoke it. Green pages include references to services' WSDL documents, which contains information on how to interact with Web services [51], [6].

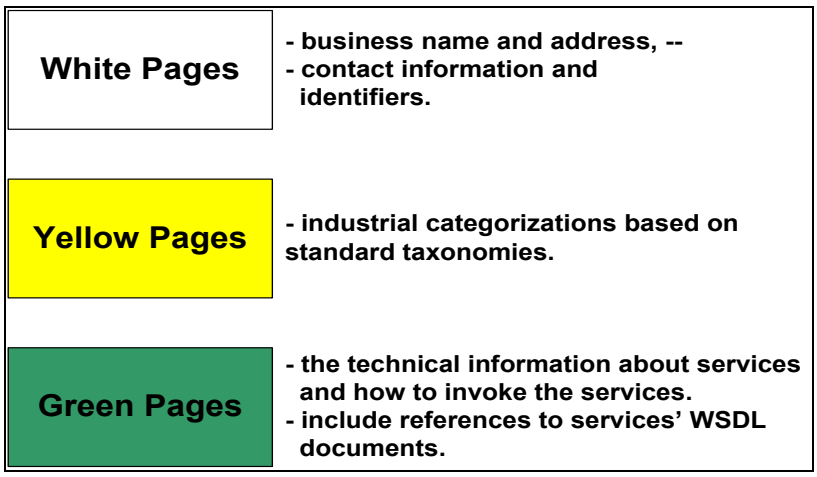

Figure 2-7 UDDI Business Registry (UBR) Components

UDDI Data Model

The basic information model used by UDDI consists of hierarchy of four basic data types. They are: business information (*businessEntity*), business-service information (*businessService*), binding information (bindingTemplate), and service specification information (*tModel*). Figure 2-8 shows the relationship among these data types.

businessEntity component encapsulates a business general information such as name, address, and contact information. *businessEntity* includes *businessServices* element which references the *businessService* component. *businessEntity* component describes different types of services offered by the company. It includes a *bindingTemplates* element which references the *bindingTemplate* component. *bindingTemplate* component provides a technical information about the services, the access point which contains the end point address, the address where to access a Web service. It contains *tModelInstanceDetails* element which references to *tModel* component. *tModels* component defines a specific information for a service. It contains *overviewDoc* element which makes *tModel* references to specific technical information which is WSDL document. *CategoryBag* is an element contains a list of industry, product or geographical classifications [52].

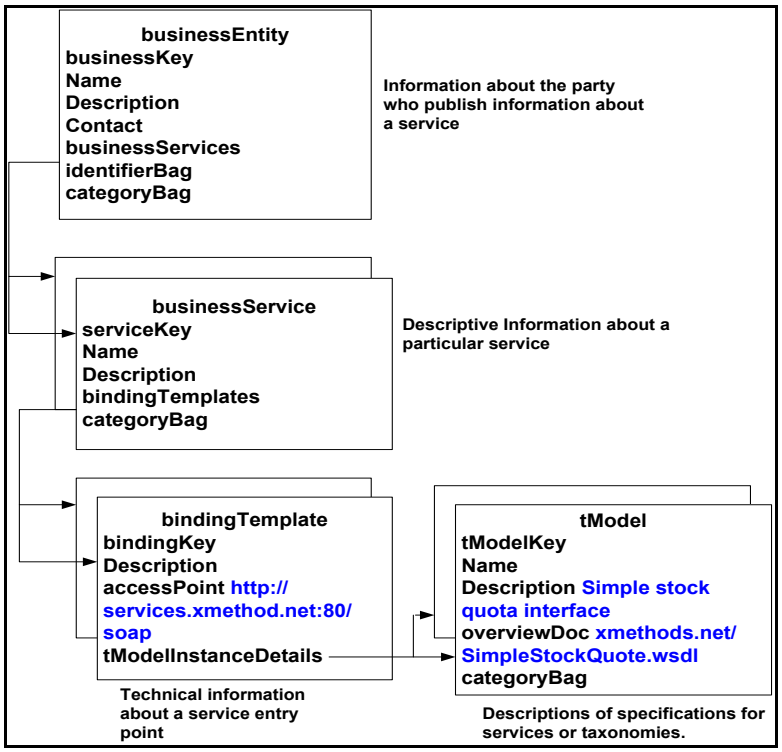

Figure 2-8 UDDI Model

UDDI API

UDDI API is SOAP-based API; all the UDDI API's methods are included within the SOAP's Body element. UDDI API methods can be divided into two categories: the inquiry methods and the publishing methods as shown in Figure 2-9.

The inquiry methods allow requester to search and browse the repository (directory), and the publishing methods allow his/her to modify the contents of the repository. The messages for the inquiry methods have a root element in the SOAP Body prefixed by *find_* or *get_*.

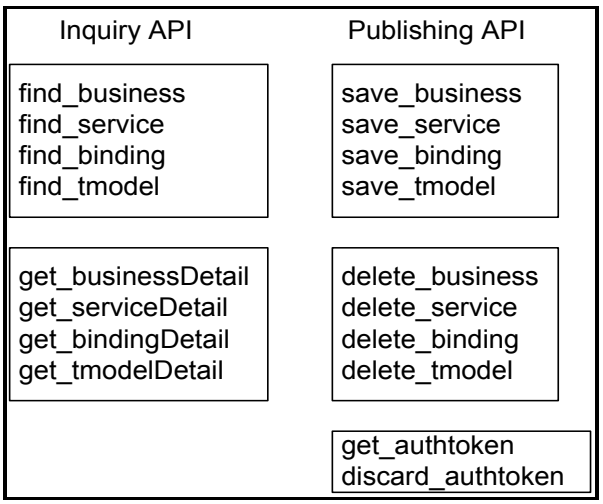

Figure 2-9 UDDI API's Methods

The messages for the publishing methods have a root element in the SOAP Body prefixes by *save_* or *delete_,* except the last two methods (*get_authtoken* and *discard_authtoken*).

The *find_* methods are for general searches, and the *get_* methods are for obtaining information about a particular record. For example *find_busines*: search *businessEntity* entities that match a specific set of criteria. Whereas *get_businessDetails*: obtains one or more specific *businessEntity* entities. The publishing API methods are for creating and updating the data within the repository by using *save_* methods. *Dele_* methods allow requester to modify and delete his/her record, to do so, he/she must include his/her authentication token (such as passport token) to prove his/her identity. An authentication token can be obtained by using *get_authtoken* method [53].

Relationship between UDDI and WSDL

Web Services Description Language (WSDL) is a mechanism used to define and describe the details regarding the communication with Web services. Universal Description Discovery and Integration (UDDI) provides a method for publishing and finding service descriptions. The UDDI data entities provide support for defining both business and service information. The service description

information defined in WSDL is complementary to the information found in a UDDI registry. The WSDL service interface definition is published in a UDDI registry as a tModel. Some of the tModel elements (such as *name* and *overviewURL*) are constructed using the information that is copied from the WSDL service interface definition. The WSDL service implementation definition is published in UDDI registry as a businessService with all relevant information copied into the businessService [54], [55]. Figure 2-10 illustrates the relationship between the WSDL and UDDI.

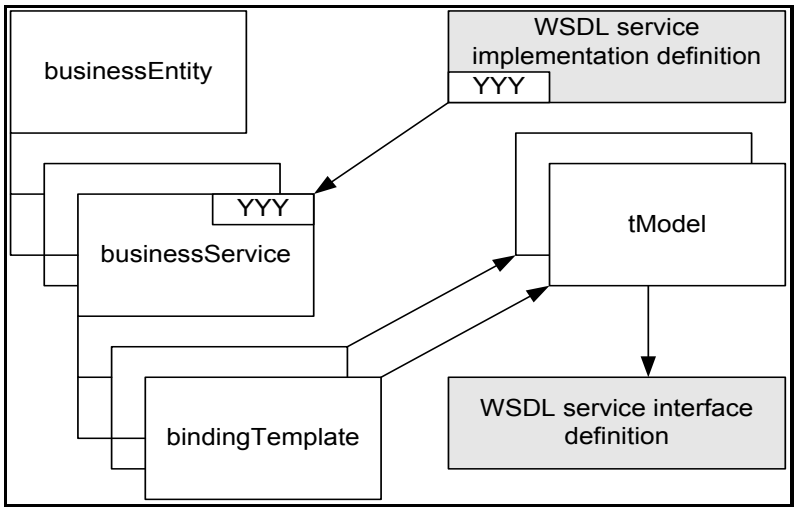

Figure 2-10 UDDI and WSDL Relationship

2.4 Technologies Used for Web Service Implementation

Web services can be implemented using Microsoft's .NET [35] and Sun Microsystems' J2EE [56] (Java 2 Platform, Enterprise Edition). J2EE and .NET are different tools with different strategies for implementing Web services [57]. J2EE and .NET are described in more details in the following sections and a comparison between them is provided below.

2.4.1 J2EE

Java 2 Platform, Enterprise Edition (J2EE) is a Java-based technology stack. It enables developers to build enterprise applications and deploy them onto any platform [58]. J2EE is based on Java programming language environment and can be run on any operating system [57]. J2EE is supported by a variety of vendors such as IBM, BEA Systems, Sun Microsystems and Oracle. The latest version of J2EE is 1.4 [56].

J2EE consists of the following components:

- **JavaServer Pages (JSPs).** Generate dynamic content for Web browsers and mobile devices.
- **ServLets:** Build control and navigation logic into J2EE applications.
- **Enterprise JavaBeans (EJBs)**. There are two types of EJB: *session beans* that model business logic and *entity beans* that model persistent data.
- **Java Connectivity Architecture (JCA)**. Enables Java enterprise applications interface with non-Java enterprise applications.
- **Java Message Service (JMS).** Provides asynchronous messaging capabilities to the J2EE platform.
- **Java Management Extension (JMX).** Manages J2EE servers and applications.
- **Java Naming and Directory Interface (JNDI).** Provides component location transparency in a clustered J2EE environment.
- **Java Database Connectivity (JDBC).** Handles all database input/output via SQL.
- **Java API for XML-Based RPC (JAX-RPC).** (J. Jeffrey Hanson ".NET Versus J2EE Web Services", 2002). Uses XML to make remote procedure calls (RPC) and exposes an API for transmitting and receiving procedure calls.

- **Java API for XML parsing (JAXP).** Allows developers to perform any Web service operation by manually parsing XML documents [59].
- **Java Architecture for XML Binding (JAXB).** Provides a fast way to create a two-way mapping between XML documents and Java objects. The JAXB compiler generates a set of Java classes containing all the code to parse XML documents based on the schema structure [59].
- **Java API for XML Web Services (JAX-WS):** It is a Java programming language API for creating Web services.It is a fundamental technology for developing SOAP based Java Web services. JAX-WS is designed to take the place of JAX-RPC in Web services and Web applications [60].
- **Web Services Interoperability Technology (WSIT):** It implements next generation Web services technologies that enable Java EE to interoperate [61].
- **XML and Web Services Security (XWS-Security):** It provides a framework within which a JAX-WS or SAAJ application developer can secure applications. Using the XWS-Security framework, developers of JAX-WS can secure their applications by configuring the request and response security policies at the level of service, port, or operation [62].

2.4.2 Microsoft's .NET Framework

Microsoft .NET is a software that enables developing applications for different environments and devices. For example, it can build XML Web services and Web applications for the Internet and can create Windows applications, server components and applications that run on any device such as PC or a mobile device. .NET integrates various applications and devices by using standards such as Hypertext Transfer Protocol (HTTP), XML and Simple Object Access Protocol

(SOAP). .NET overcomes the challenges of the software industry which is to exchange data between applications written in different languages and for different environments [63].

.NET runs on a single platform (Windows) but supports multiple languages such as Visual Basic, Visual C#, Visual J# and Visual C++, so it is a rich development platform [57].

The Microsoft .NET Framework is the infrastructure for building applications using .NET strategy. The .NET framework provides an object-oriented programming model that can build all types of applications such as Windows-based applications, XML Web services and Web applications. To create a .NET application, classes are created to define the functionality of the applications in any language supported by the .NET framework. A class written in one language is reusable by classes written in other languages. Also it can inherit classes across language boundaries because the .NET framework allows language interoperability and supports cross-language inheritance [63]. The European Computer Manufacturers Association (ECMA) standard defines the Common Language Specification (CLS), which contains the rules for language interoperability. The code written in a CLS-compliant language is interoperable with the code written in another CLS-compliant language because the code is compiled into an intermediate language (IL) code.

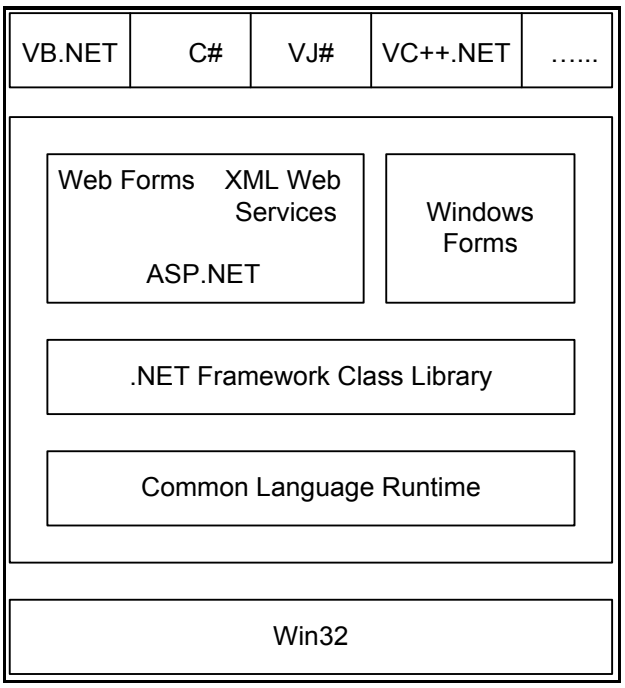

Figure 2-11 .NET Framework Components

Figure 2-11 .**NET Framework Components**shows the following components of the .NET framework.

- **Four standard CLS-compliant languages.** Microsoft Visual Basic .NET, Microsoft Visual C#, Microsoft Visual C++ .NET and Microsoft Visual J# .NET.C# is a new language for writing classes and components that integrates elements of C, C++, and Java. The compiler of these languages generates Microsoft Intermediate Language (MSIL) which makes programs written in the .NET languages interoperable [63], [64].
- .**NET applications:** .NET framework is the infrastructure for building different kinds of applications, such as console applications, Windows applications, XML Web services and Web applications. ASP (Active Server Pages) .NET is a technology for creating dynamic Web applications and Web services.
- **Common Language Runtime (CLR):** It executes programs written in any CLS-compliant language in two steps. First, a program is compiled into the

Microsoft Intermediate Language (MSIL). Second, MSIL is compiled into machine code for a specific platform. Compiling to a common format such as MSIL increases portability between platforms and interoperability between languages. The CLR is like the Java virtual machine in providing the environment in which programs execute [64].

- **Framework Class Library (FCL):** An enormous amount of pre-written of classes for creating objects such as windows and controls like buttons and check boxes, as well as handle strings, threads, network communications, Web forms, Windows services, and more. The FCL contains reusable components that programmers can incorporate into their applications, which saves them from creating new software from the scratch [6].
- **ADO+**: A new generation of ADO data access components that use XML and SOAP for data interchange [65].

2.4.3 Microsoft .NET versus J2EE

Table 2-2 shows .NET and J2EE feature comparisons as in the following:

Table 2-2 Comparison between .NET and J2EE

Feature	.NET	J2EE
Middleware Vendors	Microsoft	IBM, BEA, Sun, Oracle
Programming Language	VB, C#, J#, C++	Java
Cross-Platform Portability	only support Windows platform	complete platform portability
Web Services Support	Visual Studio .NET	JAXP
Interpreted Language	MSIL	Java Bytecode
Runtime Environment	CLR	JVM/JRE
Database Access	ADO.NET	JDBC, SQL/J

.NET

- .NET is a Microsoft platform. It runs only on a Windows platform.
- .NET supports many languages such as VB, C#, J# and C++. .
- .NET is language-independent and language- interoperability.
- .NET supports Web services through Visual Studio .NET integrated development environment (IDE).
- Source code is translated into Microsoft Intermediate Language (MSIL) which is language-neutral.
- Common Language Runtime (CLR) is Microsoft's intermediary between .NET developers' source code and the underlying hardware.
- Developers can access a variety of data sources through ADO.NET classes.

J2EE

- IBM, BEA systems, Sun Microsystems and Oracle offer a wide variety of J2EE products.
- J2EE is a platform independent that is it is portable. . It has the ability to run on any operating system [57], such as Win32, UNIX and Mainframe systems.
- J2EE supports only Java language.
- J2EE supports Web services through the Java API for XML Parsing (JAXP).
- Java source code is translated into Java bytecodes.
- J2EE offers language- level intermediation via the Java Runtime Environment (JRE) and Java Virtual Machine (JVM) which allows Java bytecode to run on any platform.
- Java Database Connectivity (JDBC) handles all database input/output via SQL.

2.5 Limitations in UDDI and Web Service Environment

This section presents the challenges in the current UDDI regarding service selection based on quality criteria and the challenges related quality matchmaking in the Web service environment.

2.5.1 Limitations in UDDI

The Universal Description Discovery and Integration (UDDI) [11], [66], [67] is proposed by Microsoft, IBM, and Ariba to provide a standard for an online registry of Web services. UDDI enables the publishing and dynamic discovery of Web services and allows developers to locate services for direct invocation or integration into new complex services. A Web service provider registers its businesses and Web services along with keywords for categorizations. UDDI describes businesses by their physical attributes such as name, address and the services that they provide. In addition, UDDI descriptions are augmented by a set of attributes called tModels, which describe additional features such as the classification of services within taxonomies such as NAICS (North American Industry Classification System) [68]. A service requester retrieves advertisements out of the registry based on keyword search [69]. UDDI suffers from the following some shortcomings:

- UDDI performs basic searching capability. The search is only done by string matching or keyword-based matching on some fields [70], [71]. Dynamic selection of adequate services involves matching of services requirements with advertised service capabilities rather than simple keywords or string [71].
- The current selection mechanism in UDDI is only based on the functional information published in the WSDL document because UDDI does not support or represent non-functional information of the Web services [33], [72]. Hence, UDDI can't search for services based on non-functional information.
- UDDI is a static registry, that is its content is specified at advertising time and can only be updated if an advertisement is replaced by a new one [32], [73].

2.5.2 Limitations in Web Services Environment

The major problems with the capability QMP in the current Web services environment are:

- Matchmaking process occurs in an open environment (Internet) which can't easily predict the quality criteria that a Web service will deliver [38].
- The service providers and service requesters have very different perspectives and it is unrealistic to expect equivalent quality specifications provided by the service providers and quality requirements provided by the service requesters to be equivalent, or even that exist a service that fulfils exactly the needs of the requester.
- Need for a common language for describing and defining the quality specification of the advertised services and the requester's quality constraints and preferences. This step is addressed by extending the Web Services Description Language (WSDL) with quality criteria as explained in Chapter 3.
- UDDI lacks the matchmaking capability essentials for selecting the right Web services. Therefore, UDDI as a service directory is important but insufficient for searching Web services and need to be complemented with advanced matchmaking facilities [73].

To address the above UDDI and Web services challenges, this thesis proposes a quality-based Web service architecture (QWSA) that extends the current Web service architecture with quality server. The core component in the quality server is the matchmaker component, which assists the service requesters to select the best available service that fulfil their preferences and satisfactions by matching between the service providers' quality specifications and service requesters' quality requirements. In addition, this project develops a quality service selection approach that assists the service requesters to select the best advertised service based on their quality preferences and requirements.

In addition, it requires a description language to express quality capabilities of services, and the specification of a matchmaking algorithm between quality specifications and quality requirements. Web Service Description Language (WSDL) is extended in this project to express quality capabilities of Web services.

In order to associates quality criteria in the Web Services Description Language (WSDL) it is required a quality classification that contains the most important quality criteria.

2.6 Semantic Web and Web Services

The semantic Web as defined by W3C as the representation of data on the World Wide Web. Adding semantics to the web involves two things: allowing documents which have information in machine-readable forms, and allowing data on the Web to be defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across applications. The objective of the semantic Web is to make electronic commerce interactions more flexible and automated [74].

The semantic Web is a participation of W3C with a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF) and Web Ontology Language (OWL). The RDF and OWL were released in 2004 by the World Wide Web Consortium as W3C Recommendations. RDF is used to represent information and to exchange knowledge in the Web. OWL is used to publish and share sets of terms called ontologies, supporting advanced Web search, software agents and knowledge management [75].

The Web is moving from being a collection of pages toward a collection of services [76]. The Semantic Web and Web Services are two visions of how to make the Web more automated use. The objective of the Semantic Web services is to describe and implement web services to make them more accessible, flexible and automated to the service requester and service provider. The semantic Web

services requires that data be not only machine readable, but also to be machine understandable.

The developers of end user applications will not to worry how to interpret the information found on the Web, as ontologies will be used to provide vocabulary with explicitly defined and machine understandable meaning [70]. DAML+OIL is an ontology language that extends RDF and which is the basis for the W3C Web Ontology Language working group's development of the OWL ontology language standard [77]. If the applications are to exchange semantic information, it need to use common language. The ontology which written in DAML+OIL and has been designed for the purpose of describing Web services, is the DAML-S ontology. DAML-S provides vocabulary for service descriptions and it aims to make Web services computer-interpretable and to enable automated Web service discovery, invocation, composition and monitoring [78].

2.7 Related Work in Quality Issues

Quality has been extensively studied in the area of computer network [79] and specially the Internet [80], and real-time computing. However, quality in the context of Web services has been a recent research activity.

The research work touches various quality issues in the Web services context. Therefore, relevant previous works on quality requirements and classification, quality Web service architecture and quality-driven service matchmaking and selection have been discussed.

2.7.1 Quality Requirements and classification

With the widespread proliferation of Web services, quality criteria will become a significant factor in distinguishing the success of service providers and to ensure that the selected Web services based on their qualities fulfil the requester expectation and requirements.

Mani and Nagarajan [82] discuss various Web service QoS requirements from the service providers perspective, that support QoS in web services: availability, accessibility, integrity, performance, reliability, regularity, and security. A QoS negotiation is used as a technique to match the needs of service requesters with those of the service providers, and using service proxy method to measure response time of the Web services.

Menasce [83] discusses the QoS issues in Web services and have to be evaluated from the perspective of the providers of Web services and from the perspective of the users of these services. These users are not human beings but programs that send requests for services to Web service providers.

Tian et al. [84], [85] propose an approach that enables the QoS integration in Web Services, and the selection of appropriate services based on QoS requirements regarding server and network performance. They describe how QoS requirements are mapped to the underlying platform and network. They also provide a Web service-QoS XML schema for the both requesters and providers to define the QoS parameters.

Seo et al. [86] present a Web service quality classification which includes the following classifications: performance, safety and cost. Performance contains response time and throughput, safety contains availability and reliability and cost contains the service cost. It presents various service levels (gold, silver or bronze) for each Web service quality aspects.

Ran [5] organizes the quality-of-service (QoS) important to Web services into categories, which are grouped into different types: QoS related to runtime, transaction support, configuration management and cost and security. Runtime related QoS contains the following aspects: scalability, capacity, performance, reliability, availability, robustness/flexibility, exception handling and accuracy. Transaction support related QoS contains integrity aspect. Configuration management and cost related QoS contains the following aspects: regulatory, supported standard, stability, cost and completeness. Security related QoS

contains the following aspects: authentication, authorization, confidentiality, accountability, traceability and auditability, data encryption and non-repudiation.

Patel et al. [87] organize the QoS parameters and classified them into the following categories: general, Internet service specific and task specific QoS parameters. General QoS parameters contain performance (throughput), performance (latency), reliability and cost. Internet service specific QoS parameters contain availability, security, accessibility and regulatory. Task specific QoS parameters contain task specific parameter.

Zeng et al. [88] propose a Web service quality based on a set of quality criteria such as availability, execution rate, execution duration, reputation and execution price. The QoS model is used to select Web services and to evaluate the QoS of composition services.

Ai-Ali et al. [31] extend the service abstraction in the Open Grid Services Architecture for Quality of Service (QoS) properties. QoS parameters are defined with respect to the three levels: application QoS (i.e., availability, reliability, accessibility); middleware QoS (i.e., memory size, number of parallel CPUs); and network QoS (i.e., bandwidth, throughput).

Gouscos et al.[89] Present a simple approach to model Web service QoS attributes and provision price, and discuss how this information can be accommodated within basic specification standards such as WSDL and exploited within the Web service deployment and application life-cycle.

Liu et al.[90] present an open, fair and dynamic QoS computation model for Web services selection. They achieve the dynamic and fair computation of QoS values of Web services through a secure user's feedback and a monitor. Their QoS model is extensible and new domain specific criteria can be added without changing the underlying computation model. They provide an implementation of a QoS registry based on their extensible QoS model.

The quality requirements are considered ,as in this thesis, from the service providers perspectives well as from the service requester perspectives in [82] and

[83]. Whereas, the quality requirements in [84] and [91] are considered from the system and network perspectives.

The quality parameters in [86], [5], [87] and [31] are classified into groups from different perspectives. In [86], the classification includes : performance, safety and cost groups. The classification in [5] includes: QoS related to runtime, transaction support, configuration and security. The classification in [87] includes: general, Internet service specific and task specific. The classification in [31] is in the Grid environment, which includes: application QoS, middlewarw QoS and network QoS. However, this thesis proposes a quality criteria classification, which organizes the most important quality criteria into four groups: Performance, Failure Probability, Trustworthiness, and Cost. Each group consists of several quality sub-criteria.

2.7.2 Quality Web Service Architecture

Because Web services can be provided by third parties and invoked dynamically over the Internet, their quality criteria can vary greatly. Therefore it is important to have a framework capturing the quality specifications provided by the providers and the quality requirements required by the requesters.

Several approaches have been represented in the literature to deal with quality of Web services.

Chen et al. [92] propose a QoS Web service architecture in which a QoS Broker acts as a mediator between service providers and service clients to make Web service selection instead of the client. The QoS Broker consists of four components: QoS information manager, QoS Negotiation Manager, QoS Analyzer and database. The Broker negotiates with QoS server(s) to make sure that the guaranteed-quality of service can be provided to the clients. The key QoS attributes considered in [92] are Web services response time, cost, network bandwidth, and service availability. However, the proposed quality-based web service architecture (QWSA) differs from the aforementioned architecture in that

it does not use negotiation to select the desired service, but it selects the best available Web services by using the quality matchmaker component in the quality server and use mathematical technique for matching the quality specifications against the quality requirements and without requiring negotiation.

Seo et al. [86] propose Web Service Quality Broker Architecture, which helps service requester to find the optimal Web service. They described negotiation process by using Multi-Attribute Utility Theory (MAUT) on the basis of quality information of both sides (service requester and service provider) participating in negotiation. Quality model is proposed by classifying the quality attributes into performance, safety, and cost aspects.

Ran [5] proposes a new Web Services discovery model in which the functional and non-functional requirements (i.e., quality of service) are taken into account for the service discovery. A QoS certifier is introduced in this model that certifies the QoS claims given by the providers and verifies these claims for the clients. An extension to UDDI's data structure types is proposed for implementing the proposed discovery model.

Serhani et al. [93] present a broker-based architecture for QoS management for Web services. They propose a QoS broker which is used as a third party Web service published in UDDI registry. It is invoked when a user requests a Web service with QoS requirements. The role of the QoS broker is to support QoS provisioning and assurance in delivering Web services. It introduces a new concept, called QoS verification and certification, which is used together with the QoS requirements in the selection process of Web services.

Yu and Lin in [94] present a QoS-Capable Web Service Architecture (QCWS) in which a QoS broker acts as a mediator between service providers and clients. The QoS server collects QoS information about servers, makes select decisions for clients, and negotiates with servers to get QoS commitments. The non-homogeneous resource allocation algorithm (RQ) is used to allocate different amounts of resources to different clients according to their requirements.

Chen et al.[95] propose UX (UDDI eXtension), a system that is QoS-aware and facilitates the federated discovery for Web services. The QoS feedback from service requesters are used to predict the service's performance. UX server supports wide area discovery across domains. The UX server's inquiry interface conforms to the UDDI specification. A discovery export policy is proposed that controls how the registered information is exported to UX servers and requesters.

Patel et al.[87] propose a QoS oriented Framework, called WebQ, that is able to conduct the adaptive selection process and provides binding and execution of Web services for the underlying workflow. They have designed a QoS model for Web service selection, binding, and execution. They develop a set of algorithms to compute QoS parameters and implement them using a rule-based system. QoS model selects dynamically the best available services and executes these services to maximize the overall QoS. The QoS parameters are classified into three categories: general, Internet service specific, and task specific.

Menasce in [96] describes a framework called Q-application and Q-component for QoS-aware software components (distributed applications)., and focus specifically on QoS requirements such as performance, availability and security for such framework performance. A Q-application can discover the Q-components that provide given services and a QoS negotiation between the Q-application and Q-component occurs and if the negotiation is successful then the Q-component becomes part of the Q-application. However, no methods are mentioned to describe how to discover the services.

ShaikhAli et al. in [30] implement UDDIe- an extension to UDDI which supports the notion of "blue pages" to record user defined properties associated with a service and to enable search on other attributes of a service by extending the *businessService* class in UDDI with *propertyBag* and to discover of services based on these.

Different approaches have been introduced in order to extend the current Web service architecture with quality capabilities. A QoS Broker has been introduced

as a mediator between the service requesters and providers, and it is used in order to select the best service in [92], [86], [94] and [95]. The negotiation process as in [96] is used to select the best service. Also, the However, the QoS Brokers are not well defined. These are no information about how the QoS brokers discover and select the optimum Web services.

Another approach using QoS certification concept in both [5] and [93], but with different functions. In [5], the QoS certifier extending the original UDDI model and verifies the QoS claims for a Web service before registration. Whereas in [93], QoS certifier is a module in the QoS broker for certifying Web services and their provided QoS.The QoS certifier which introduced in [5] is not well defined; it does not describe the details of the certification process as in [93].

The current UDDI in [30] is extended with *propertyBag* element in the *businessService* class that enables the service providers to publish their service with quality aspects and enables the requesters to discover the services based on quality aspects.

From the previous approaches it can't find a comprehensive solution for selecting the best available Web service based on quality criteria. The Broker functions are not well defined and no details for the service selection. This thesis proposes a quality-based Web service architecture (QWSA), to bridge the gap between the service requester's quality requirement and the service providers' quality specifications. This architecture incorporates a quality server that facilitates and assists the service requester to discover and select the best available Web services. The core component of the quality server is a quality matchmaker, which selects the best service based on a mathematical model.

2.7.3 Quality Service Matchmaking and Selection

There are several research activities related to matchmaking, discovery and selection work which are based on, semantic and QoS characteristics as in the following:

Facciorusso et al. [73] propose a matchmaking process in the context of Web services by using Web Services Matchmaking Engine (WSME). WSME is a Web service supplied as part of the IBM Web Services Toolkit (WSTK) [97]. The WSME matchmaking process is a two ways or symmetric process where each party (customer or provider) submits a description of itself and the requirements of the other part. The matchmaking process evaluates the demands of each party against the descriptions of the other parties by using rules, which allows both parties to select each other. Also the paper proposes the drawbacks of UDDI. UDDI is limited in search capability and the search is asymmetric which means that only customers have the ability to express their requirements of the service and its providers, but not vice versa. UDDI is a static directory that is its contents is specified at advertising time and can only be updated if an advertisement is replaced by a new one. UDDI also lacks the matchmaking capability which is essential for selecting the right Web services. So, UDDI are important as a directory service but insufficient for selecting the right Web services and need to be complemented with advanced matchmaking facilities.

Ran [5] proposes a model for Web service discovery with QoS by extending the current UDDI model with QoS information. But service research and selection are still done by human clients. This is not desirable if thousands of services are available for selection. Searching and finding the most suitable service that match the requester's QoS requirements may be better performed by an automated system. However, this thesis develops a quality service selection system (QSSS), which enables the requester to select the best service automatically.

Farakas and Charaf [98] propose a software architecture to provide QoS-enabled Web services by adding a QoS broker between clients and service providers to discover the QoS aware services in UDDI. However, there is no detailed information about the functionality of the QoS broker.

Balke and Wagner [33] propose a cooperative discovery algorithm for selecting a suitable services by using an ontology-driven approach DAML-S. Also, the paper

mentioned the UDDI shortcomings: UDDI is limited to keyword matching and does not support any inference to relax descriptions associated in user preferences or ontologies.

The above paper based on semantic matching by using DAML-S semantic Web services framework and the matching doesn't address the QoS issues. However, our project will propose matchmaking algorithm using mathematical techniques and based on requester's QoS preferences.

Wang and Stroulia [99] propose a flexible service discovery method which based on information retrieval and WSDL structure matching. An information retrieval method uses vector space model to identify most similar service description files and to order them according to their similarity. Then, a WSDL structure-matching algorithm is used to refine and assess the quality of the candidate service set. The WSDL structure matching includes matching the structure of the operations' input and output messages, and matching the data types of the objects communicated by these messages. Also, the paper mentioned the drawback of UDDI specific QoS properties. Also, there is no method explaining how to rank and select the best Web services.

Maximilien and Singh [38] propose a comprehensive agent-based trust framework for service selection in open environment. The authors introduce a policy language to capture service consumer's and provider's profiles. They introduced QoS ontology as a specification which enables matching services semantically and dynamically. The semantic matchmaking allows the service agent to match consumers to service using the provider's advertised QoS policy for the services and the consumers' QoS preferences. The provider policy and consumer preferences are expressed using the concepts in the QoS ontology (QoS model). The service selection is based on user preferences and business policies, and considers the trustworthiness of service instances. So, their approach enables applications to be configured dynamically at run time to select the best services with respect to each participant's preferences.

Sycara et al. [100], [101] present a flexible and efficient matchmaking process that uses LARKS (Language for Advertisement and Request for Knowledge Sharing) which is a language for agent advertisements and requests. The LARKS matchmaking process performs both syntactic and semantic matching. The service specification is written in the concept language ITL (Information Terminological Language). The matchmaking process uses five different filters: context matchmaking, profile comparison, similarity matchmaking, signature matchmaking and constraints matchmaking. Different degree of matchmaking can result from using different combinations of these filters.

Ouzzani and Bouguettaya [102] propose a novel infrastructure that optimizes query facilities for Web services. They propose a matchmaking process which matching virtual operations to concrete operations. The query model determines the best service is based on QoS parameters "QoWS", service rating, and matching degrees. However, the authors do not cover about the matchmaking process related to QoS parameters.

Zhou et al. [103] propose a QoS ontology called DAML-QoS ontology as a complement for DAML-S ontology to provide a better QoS metrics model. It is designed for the matchmaking purpose. Matchmaking algorithm for QoS property constraint is presented and different matching degrees are described.

However, the above paper provides a novel DAML-QoS ontology which is based on DAML+OIL layer instead of XML layer. A DL reasoning is used to match requester's *QoSProfile* to advertisement *QoSProfile* according to the matching degrees (subsume, exact, plugIn, intersection, and disjoint). DAML-S is a DAML+OIL (an ontology language used in the Semantic Web) ontology for describing Web services. DAML-S is extended by quality of service metrics description for service discovery to meet user needs. Well this thesis uses WSDL description language instead of DAML-S, and WSDL is extended with QoS criteria specification. The matchmaking process has four stages or filters: interface

matchmaking, quality criteria type matchmaking, quality criteria value matchmaking and mathematical matchmaking.

Pilioura et al. [104] propose an infrastructure for web service publication and discovery (PYRAMID-S), which addresses the UDDI limitations by combining the technologies of Web Services, Semantic Web and Peer-to-Peer Networking. The main contribution of this infrastructure is that the Web service publication and discovery based on syntactic semantic information as well as on QoS characteristics in order to enable result ranking and service selection.

Al-Ali et al. [31] propose a framework in service-oriented Grid. The advertised services are discovered based on QoS criteria by using service level agreement. WSDL and UDDI are extended by QoS properties. The matchmaking broker matches the queries with advertised services based on QoS properties.

Li and Horrocks [70] propose a matchmaking process which based on DAML-S semantic Web ontology and a Description Logic (DL) reasoner to compare ontology based service descriptions.

Zeng et al. [88] present two service selection approaches; local optimization and global planning. A Simple Additive Weighing technique is used to select an optimal Web services. The users express their preferences regarding QoS by providing values for the weights. They propose a simple QoS model using the examples of price, availability, reliability and reputation.

Liu et al.[90] present an open, fair and dynamic QoS computation model for Web services selection. They achieve the dynamic and fair computation of QoS values of Web services through a secure user's feedback and a monitor. Their QoS model is extensible and new domain specific criteria can be added without changing the underlying computation model. They provide an implementation of a QoS registry based on their extensible QoS model.

Fedosseev in [105] present the *global planning approach* which used to optimally select component services during execution of a composite service. The approach is based on quality-of-service (QoS) characteristics of services Different types of

quality metrics have been introduced such as QoS: system, QoS: task, quality-of-experience (QoE), and quality-of-business (QoBiz).

Some of the matchmaking and selection technique are general and not consider the quality issues as in [73], [100], [101] and [99].

Most of the previous research on service discovery matchmaking and selection is based on syntactic and semantic service characteristics. The syntactic information comprises the service name and a short textual service description. The semantic information refers to machine-understandable meaning to the concepts of the service description. However, rarely researches enriched their service discovery, matchmaking, and selection techniques with quality aspects as in [73], [100], [101] and [99]. Most of the related quality matchmaking based on either semantic as in[33], [38], [106], [104] and [70] or computation as in [88], [90] and [105].

This thesis proposes quality matchmaking selection technique that is based on the mathematical model. The Analytical Hierarchy Process (AHP) is used to calculate the quality criteria weight based on the requester preferences. The Euclidean distance is used to calculate the distance between the quality requirements and the quality specifications. The service associated with the minimum distance is the best service to select..

2.8 Summary

This chapter has introduced an overview about Web service architecture and its standards. This chapter shows that Web services technology offers many benefits and provides more advantages over the distributed-computing technologies as the Web service is interoperable which has the ability to communicate and share data with software from different vendors and platforms.

But Web services technology also has some challenges. The Web services standards are still immature and under development. The UDDI standard is a registry database and service discovery engine and allows requester to look for Web services based on their functionality but not quality information. WSDL is

an XML format for describing Web services; it does not address issues related to the description of quality aspects of a service. In addition, selecting services regarding its quality over open environment is difficult and challenging because of the dynamic nature of the quality criteria that can't easily be predicted and it is not easy for the service requester to select the best service of the same functional properties with different quality criteria information.

To address the aforementioned challenges, this project proposes quality Web service architecture (QWSA) that extends the current Web service architecture with quality server. The quality server consists of four main components: quality Manager, quality Matchmaker, quality Report Analyzer, and quality Database. The main purposes of the QWSA architecture are to:

- Enhance the current UDDI role by enabling service publishing and discovering based on quality criteria.
- Matches the quality specifications of the advertised Web services against the quality requirement that specified by the service requester.
- Assists the service requester to choose the best available service based on his/her quality requirements and preferences.

Also, this project develops a quality matchmaking process (QMP) that assists the service requester to select the best advertised service based on his/her quality preferences and requirements by matching between the service providers' quality specifications and service requesters' quality requirements.

The QWSA architecture and the QMP will be discussed in details in the coming chapters.

Chapter 3 Quality Definition

3.1 Introduction

Web services quality is an important factor from the requester point-of-view because it differentiates similar services offered by different service providers.

Section 3.2 gives the definition of quality criteria in Web services syntax.

Section 3.3 formulates a conceptual quality criteria classification that consists of four quality criteria groups: Performance, Failure Probability, Trustworthiness, and Cost. Section 3.4 extends the current Web Service Description language (WSDL) with the quality criteria classification by adding a new element tag called <QualityCriteria>.

3.2 Quality Criteria in Web Services

Web services technology is becoming increasingly popular and more businesses are planning to build their future solutions on it. Future business systems require integration of business processes, business applications, and Web services over the Internet. Delivering quality of the services is a critical and significant challenge because of the dynamic and unpredictable nature of business applications and Internet traffic. Due to this rapid growth, quality of the service is becoming a significant factor and playing an important role for the success of this emerging technology.

3.2.1 Quality Concept

Quality criteria have different definitions in different domains. However, in the Web services context, quality criteria is defined as a set of non-functional criteria such as availability, performance and reliability that impact the performance of

Web services [107]. Given a set of quality criteria, the aim of Web services is to match the needs of service requester with the published services [82], [1].

Quality is the measure of how well does a particular service perform relative to expectations, as presented to the user. The type of quality may be relative to the expectations of the requester who requests the service or may be relative to the expectations of the service provider who offer/deliver the service. It determines whether the requester will be satisfied with the service delivered, that is, the quality is meeting requirements. Quality can be expressed in user perceptions in a number of parameters, which have either subjective or objective values. Objective values can be measured automatically, whereas subjective can be measured by involving the humans. Quality is dynamic which means that the requester and provider can modify their requirements and offers' criteria to eliminate the gap between them [108].

The international quality standard ISO 8402 [109] describes quality as "*the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied need*". This thesis defines quality in Web service environment as a set of non-functional attributes that both service provider and service requester can be able to specify quality criteria related statements to enable quality criteria aware service delivery, service lookup, service selection and service consumption.

3.3 Quality Criteria Classification

The service providers and service requesters have different perspectives and that can't to expect equivalent quality specifications provided by the service providers and quality requirements provided by the service requesters. Therefore, a quality criteria classification is required in order to capture the descriptions of quality criteria from requester's perspective as well from provider's perspective that are applicable to all Web services. In addition the quality classification is required in the selection process to enable the requester to select the best service based on

his/her quality requirements. The quality requirements consider both the Web services quality and their corresponding services or products quality. Section 1.1.1 defines the notion of the Web services and the services they are provided.

The quality criteria classification in this thesis is similar to the quality classification in [86], [5] and [87] in that they classify the quality criteria into groups with different perspectives. The quality classification in [86] includes three groups: performance, safety and cost. Performance contains response time and throughput, safety contains availability and reliability and cost contains the service cost. The quality classification in [5] organizes the most important quality-of-service important to Web services into four groups: QoS related to runtime, transaction support, configuration management and cost and security. The quality classification in [87] classifies the QoS parameters into the following groups: general, Internet service specific and task specific. General QoS parameters contain performance (throughput), performance (latency), reliability and cost. Internet service specific QoS parameters contain availability, security, accessibility and regulatory. Task specific QoS parameters contain task specific parameter.

However, this thesis proposes a quality criteria classification that organizes the most important quality criteria into four groups: Performance, Failure Probability, Trustworthiness, and Cost regarding its characteristics and includes generic sub-criteria. The generic sub-criteria are applicable to all Web services, reusable across domains (e.g., business and scientific) and can benefit all service requesters. Quality criteria classification as shown in Figure 3-1 is extensible as in [87], in which the new criteria can be added without fundamentally altering the mathematical mechanism and the service selection techniques built on top of the classification [90], [110]. Mathematical mechanism and service matchmaking and selection technique will be discussed in Chapter 5.

The quality criteria groups have the following characteristics:

- Each group has a set of metrics, dimensions or parameters which capture subjective or objective values. Objective values can be measured automatically such as the response time, whereas subjective can be measured by involving the humans such as the reputation.
- Some of the criteria could be negative that is, the higher the value, the lower the quality. This includes criteria such as response time and service price. Other criteria are positive that is the higher the value, the higher the quality. This includes criteria such as availability and reputation [88].
- Quality criteria are deterministic and non-deterministic. Deterministic indicates that the value of quality criterion is known when a service is invoked, for example, the service price. The non-deterministic is for quality criterion that is unknown when a service is invoked, for example, execution time. For deterministic criteria, the service providers advertise them in the UDDI registry. Whereas, the non-deterministic quality criteria are computed during the runtime service execution [90].

The quality criteria parameters have related properties or elements q*value* (range value), and *unit* for both quality specifications provided by the service providers and quality requirement provided by service requester. In addition to the aforementioned elements, quality requirement has also *weight* criteria to express different requesters' demands and preferences. The quality criteria elements are described in the coming section.

The quality classification is implemented by developing a quality service selection system (QSSS) that enables the service requester to specify his/her quality requirements. QSSS system is described in Chapter 6.

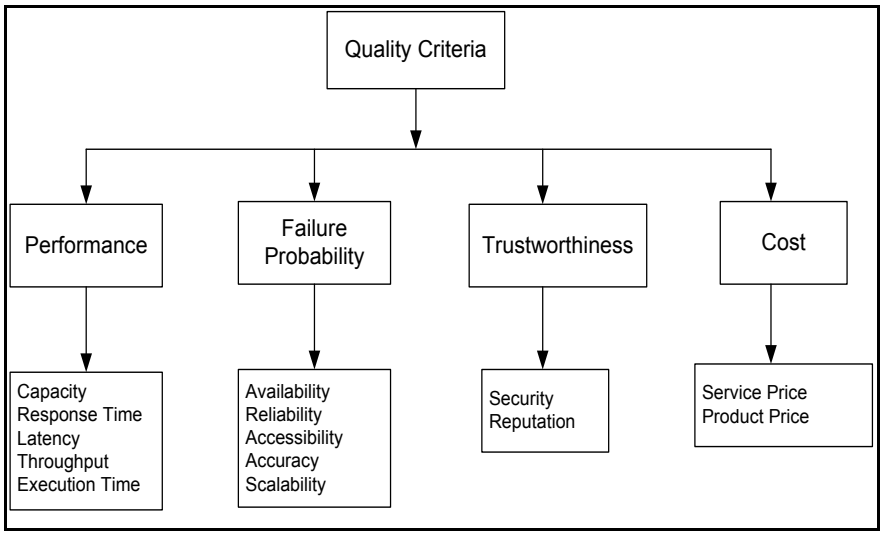

Figure 3-1 Quality Criteria Classification

The four groups and their sub-criteria are described below.

Performance

The performance of a Web services measure the speed in completing a service request. It can be measured by:

Capacity. The limit of concurrent requests that the service support for guaranteed performance.

Response time. The maximum time that elapses from the moment that a web service receives a SOAP request until it produces the corresponding SOAP response [89]. It is positively related to capacity [5]. Response time is defined as the total time needed by the service requesters to invoke the service. It is measured from the time the requester initiates the invocation to the time the requester receives the last byte of the response [103].

Latency. The round-trip time between sending a request and receiving the response [82].

Throughput. The number of Web service request completed at a given time period. It is the rate at which a service can process requests. Throughput is related negatively to latency and positively to capacity [111].

Execution (processing) time*:* The time taken by a Web service to process its sequence of activities [111].

In general, high performance Web services should provide higher throughput, higher capacity, faster response time, lower latency, and lower execution duration.

Failure Probability

The failure probability is the probability of a Web service being incapable to complete a service SOAP request within the maximum response time corresponding to this request [89]. The failure probability is composed of:

Availability. It is related to the availability of the Web services and the availability of their corresponding services or products.

Web service availability: The Web service is available when it is ready for immediate invocation [111]. Associated with availability is Time-to-Repair (TTR) which represents the time it takes to repair the Web service [82, 112]. The availability A(s) of a service s is the probability that the Web service is accessible or the percentage of time that a Web service is operating [83, 89]. For example, Amazon E-Commerce Service (ECS) is available when the requester enables to access it and searching for products such as books.

Service or Product availability: It is available when the product is ready to be used or invoked. For example, after retrieving a result about books when searching ECS, a book is available when the requester can buy it immediately.

Reliability*:* It is the probability of a service to perform its required functions under stated conditions within a maximum expected time interval [5]. It refers to the assured and ordered delivery for messages being sent and received by service requesters and service providers [82]. It can be measured by: Mean time between

failure (MTBF), Mean Time to Failure (MTF), and To Transition (MTTT) [5]. Reliability is closely related to availability.

***Accessibility**:* It is the capability of serving the Web Service request. The Web service might be available but not accessible because of a high volume of requests [82]. Accessibility can be represented by the following formula: $P_{accessibilty} = P_{availabilty}$ at Time T=t [87].

Accuracy: It is the amount of errors produced by the service during completing of the work [5].

Scalability: It is the capacity of increasing the computing capacity of service provider's computer system and system's ability to process more operations or transactions in a given period of time [5].

Trustworthiness

Trust in general is a rational concept involving the trusted and the trusting parties. For example, on the eBay Web site, eBay is a trusted authority who authenticates the sellers in its auctions and maintains their ratings. However, eBay would be unable to authenticate parties who were not subject to its legal contracts covering bidding and selling at its auctions [113]. IBM and Microsoft proposed WS-Trust specification that build on WS-Security to provide a framework for requesting and issuing security tokens for establishing trust relationship [114] .

The trustworthy of service providers affects the requester's service selection decision. The requester selects the services from providers of the highest level of trust [38].

Web services trustworthiness can be achieved when the selected Web services components fulfil its requester needs or requirements (i.e., functional and non-functional) [115].

Web services trustworthiness can be measured by:

***Security**:* it represents the measure of trustworthiness. With the increase in the use of Web services which are delivered over the public Internet, there is a grown concern about security. The Web services provider may apply different approaches and levels of providing security policy depending on the requesters needs.

IBM and Microsoft proposed a WS-Security [114] standard which a family of protocols that enhances SOAP [2] messaging technique to solve the problems about the quality of protection for Web services such as: authentication and authorization of users, message integrity, and message encryption.

Security for Web service can be provided by the following mechanisms:

- **Transport- Level Security**. Secure Socket Layer (SSL) [116] is the most widely used transport security data-communication protocol. SSL is a protocol developed by Netscape for transmitting private documents via the Internet. SSL provides authentication (the communication is established between two trusted parties), confidentiality (the data exchanged is encrypt), and message integrity (the data is checked for corruption). SSL support transport security between two SSL-enabled parties. For example, when an application invokes Web services A for purchasing and Web services B for shipping, then two SSL sessions is needed. Another protocol for transmitting data securely over the World Wide Web is Secure HTTP (S-HTTP). SSL creates a secure connection between a client and a server, over which any amount of data can be sent securely, whereas S-HTTP transmits individual messages securely. SSL and S-HTTP are complement each other and have been approved by the Internet Engineering Task Force as a standard [117].
- **Authentication**. Determining the identity of the sender [118] Service requesters need to be authenticated by the service provider before sending information. Standard Web technologies using passwords, certificates, Kerberos, LDAP, and Active Directory can be used to authenticate service requesters.

- **Authorization**. Determining if the sender is *authorized* to perform the operation requested (explicitly or implicitly) by the message [118]. That is, what the requester are permitted to access?
- **Integrity**. Message integrity is protecting the message content from being illegally modified or corrupted [114]. Data integrity is to protect the data in a database from an unauthorized insertion, modification or destruction.
- **Confidentiality**. Confidential information is to ensure that information/data is protected against the access of unauthorised principals (users or other services) [119]. Also Confidential message is to protect the message content from being intercepted [114]. WS-Security specification provides a means to protect a message by encryption and /or digital signing [114].
- **Accountability**. The provider can be hold accountable for their services [5].
- **Traceability and Auditability**. The possibility to trace the history of a service when a request was serviced [5].
- **XML Data encryption**. XML data encryption used to satisfy the high-level security principle of *confidentiality* for Web services [120], [121]. XML encryption allow encryption of digital content, such as Graphical Interchange Format (GIF) images, Scalable Vector Graphics (SVG) images, or XML fragments [122]. Encryption of an XML document can be partial, that is encrypt parts of an XML document while leaving other parts open The XML Encryption specification describes how to use XML Signature with XML Encryption so that trusted parties can selectively encrypt and sign parts of documents [123].
- **XML Digital signature.** It is a standard for securely verifying the origins of messages. The purpose of an XML signature is to associate a private key with referenced data to guarantee the sender's authentication and thus assuring that the data is really coming from a trusted originator [123], [124]. XML digital

signature is used to satisfy the high-level security principle of *integrity* [121] and can be used for validation of messages and for *non-repudiation* [122].

- **Non-Repudiation**. It proves the identity of the originator of the SOAP message, and to prove the fact that they sent the message [122].

***Reputation*:** it is the measure of trustworthiness of a service, based on the requester experiences of using the service. Different requesters may have different opinions on the same service. The reputation can be defined as the average ranking given to the service by the requesters. The value of the reputation is computed using the expression $q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the requester ranking on a service's reputation, n is the number of times the service has been graded. Usually, the requesters are given a range to rank Web services, for example, in Amazon.com, the range is [0,5] [110].

The notion of reputation is tightly bound to history and time. An approach to associate timestamps with attribute values that allowing the reputation rating to weight attributes depending on their ages.

Cost

It is the cost charged by the service provider entity to the service client entity fro a request that is successfully responded [89]. The successful response is the response produced within the maximum response time defined for this type of request. The cost value is measured by:

Web Service Price It is the amount of money that the service requester has to pay for using or invoking a Web service such as using Amazon E-Commerce Service (ECS) to search for products.

Product Price: It is the amount of money the service requester has to pay to the seller to buy a product such as a book after searching the ECS Web service.

The total cost is calculated by:

Total Cost=Web service price+ product price

3.4 Quality Extension to WSDL and UDDI

Different requesters may have different preferences or requirements on qualities as well as different service providers may offer different quality specifications for the same offered services. It is important to represent quality criteria from the perspective of service requesters' preference as well as from service provider perspective [90]. Quality criteria from requester perspective is that the service specifications of the WSDL can be extended with quality statements which describe the required qualities associated with the service required by the requester [82]. Whereas quality criteria from provider perspective is the quality statements that describe the offered qualities associated with the service offered by the service provider [82] . This thesis focuses on the quality criteria from the requester perspective.

The requester needs to specify his/her quality requirements in the Web Services Description Language (WSDL). The WSDL does not support quality issues, so it needs to be extended with quality criteria. In order to associate quality criteria in the WSDL it is required a quality classification that contains the most important quality criteria. This classification is described in Chapter 3.

Various approaches were proposed to enable standardized quality specification for Web services. Tosic et al [125], [126] present a special-purpose language Web Service Offerings Language (WSOL) dedicated to formally specifying QoS attributes of Web services, as well as other management information (such as access rights and pricing policies), on the top of the WSDL templates. DAML-S (DAML-Services) [78] is a semantic description language of Web services, including specification of functional and some QoS constraints. IBM's Web Service Level Agreement (WSLA) framework [127], [128], [129] is an XML specification of SLA which enable the specification and monitoring of QoS-aware

Web services by applying an electronic SLA. Maximilien and Singh [108] use XML policy language (WS-Policy) to specify service consumer's QoS preferences or policies and service provider's quality advertisements. DAML-QoS ontology in [103] is complement to the semantic description ontology DAML-S [78] and has been developed to design patterns for the formal specification of various types of constraints and QoS metrics. All these efforts are not focusing solely on quality criteria specifications, but rather on various facets of Web services in order to support the modelling and management of service level agreements (WSOL and WSLA specifications), service invocation policy (WS-Policy specifications), as well as semantic annotation (DAML-QoS specifications).

But, this thesis accommodates quality criteria classification within existing Web services core specification standards that is Web Service Description Language (WSDL) and Universal Description Discovery and Integration (UDDI). The key idea behind this work is to accommodate service description with quality criteria and to enhance the service matchmaking and selection process based on quality criteria [31].

3.4.1 Extended WSDL

WSDL is the current standard for specification of Web services. WSDL documents can be used to register services with the UDDI registry. There are two kinds of documents that are used while registering a service [55]. The first is known as the *Service Interface Document* that provides an abstract definition of a Web service and omits implementation details such as port address, communication protocol, etc. The other document is the *Service Implementation Document* that contains a description of a service that implements a service interface. The relationship between WSDL and UDDI is described in section 2.3.2

Although WSDL is an XML format for describing Web services, it does not address issues related to the description of quality aspects of a service [130]. In this thesis, WSDL is extended to accommodate quality criteria of the proposed

quality criteria classification. The quality criteria extension is made in the *Service Implementation Document* part as extended in [89], [131]. Because WSDL is an XML based language, the proposed quality classification is implemented using XML Spy in order to design Quality Criteria XML Schema (see appendix A for details). XML Spy is the industry standard XML development environment for designing and editing professional applications involving XML, XML Schema, XSL/XSLT, and other XML-based technologies. XML Spy Home Edition [132] allows creating and editing XML Schema but not allow creating, editing, visualizing, and validating any WSDL file. XML Spy Home Edition is selected because it is free application and suitable for students. XML Spy Enterprise Edition [133] allows editing WSDL but it is expensive to buy. Then a new <QualityCriteria> element is augmented within the WSDL <service> element. This extension is explained in the following figures.

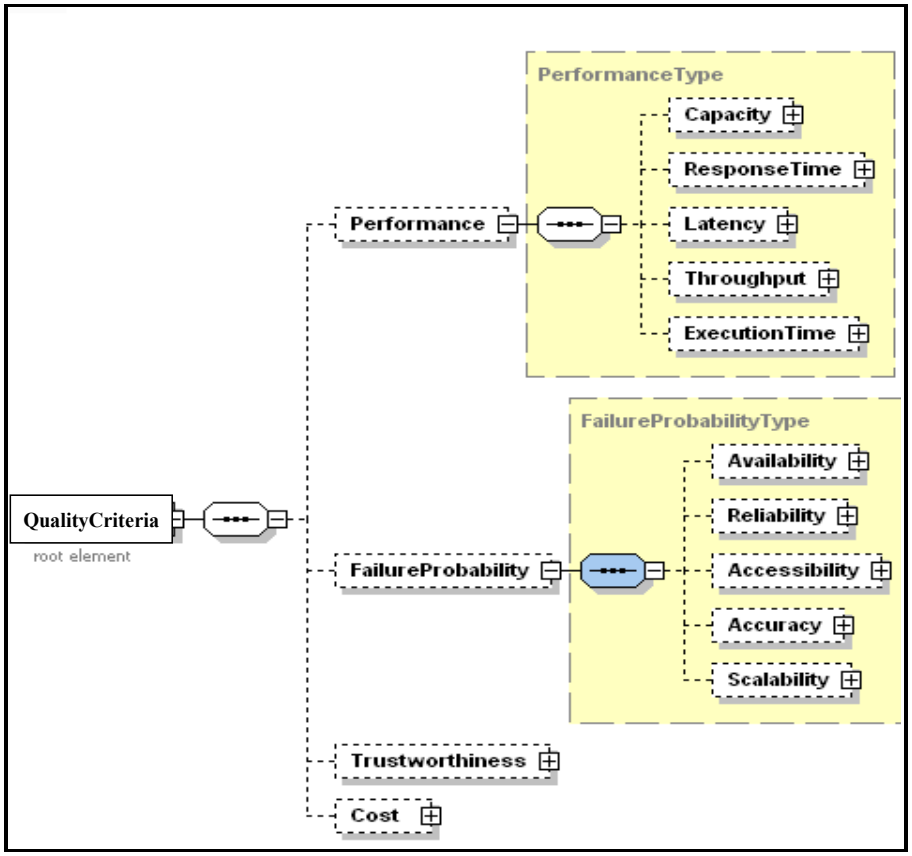

Figure 3-2 Screenshot showing sub-criteria elements for Performance and Failure Probability in Quality Classification

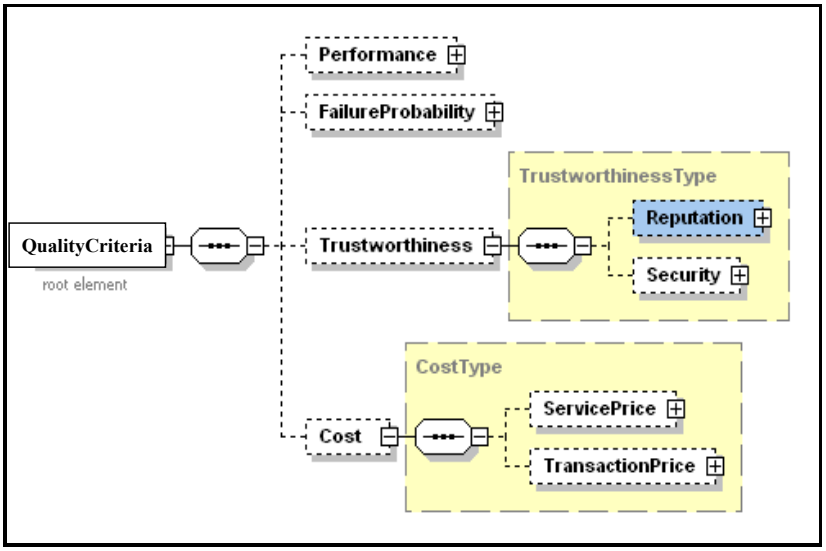

Figure 3-3 Screenshot showing sub-criteria elements in Trustworthiness and Cost Criteria in Quality Classification

Figure 3-2 and Figure 3-3 show quality sub-criteria of each quality criteria group (Performance, Failure Probability, Trustworthiness, and Cost).

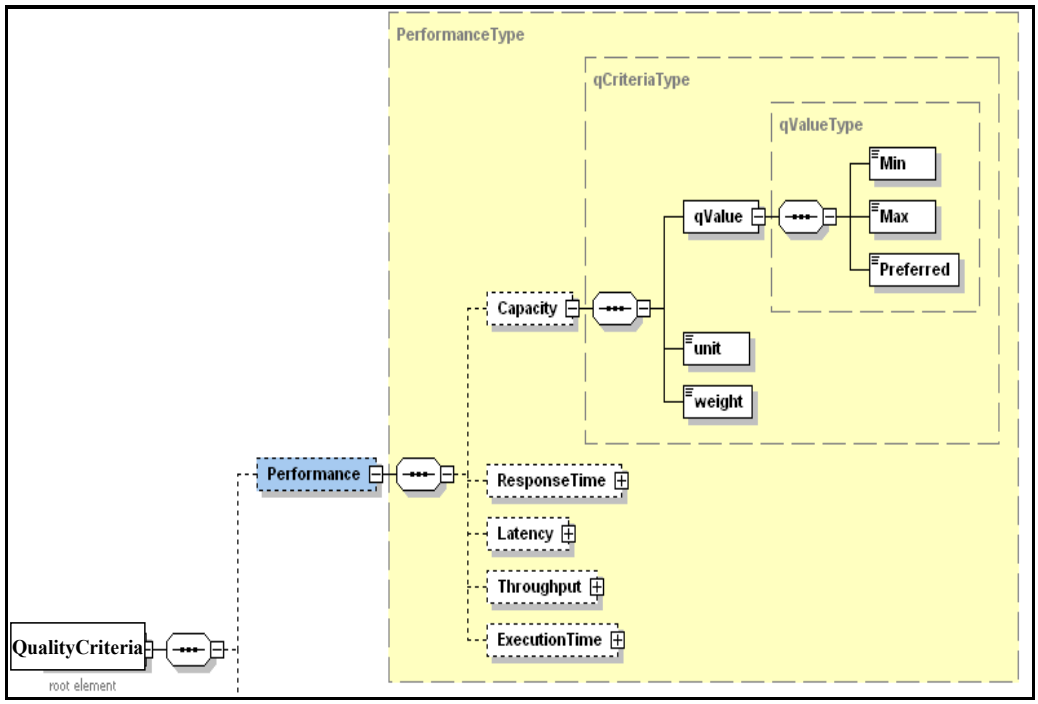

Figure 3-4 Screenshot showing properties for each Sub-Criteria element

Figure 3-4 shows the properties or child elements (*qValue*, *unit*, *weight*) for each sub-criterion. *qValue* has the value of sub-criteria, *unit* has enumerator values (*Msec*, *Percentage*, *Request/sec*, *Pound* and *None*), weight has value range between [0,1] and the default value is 1. *qvalue* includes further child elements (*Min*, *Max*, *Preferred*) and attribute called *qlevel*. *Min, Max, and Preferred* has the minimum, maximum and preferred values from the requester point of view. *qlevel* has enumerator values (*High*, *Medium*, and *Low*) which is the level of importance associated with every quality sub-criteria. For example, *High* value regarding the sub-criteria *Availability* is between [90, 99], whereas for *Reputation* is between [4, 5], these levels will be described in Chapter 6. The above elements and child elements are shown in Appendix A.

Figure 3-5 shows an example of quality requirements by extending Amazon Web service WSDL with Quality Criteria XML Schema. Amazon Web service WSDL document can be retrieved from the URL: http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl. Amazon Web Service or Amazon E-Commerce Service (ECS) [134] (see

Appendix D for details) provides many request operations to look up Amazon products. Two request operations are selected: *ItemSearch* and *ItemLookup.*

WSDL as explained in Section 2.3.2 consists of two primary parts: the *services interface definition* that contains *message*, *portType* and *binding* elements as shown in the first part of Figure 3-5; and the *service implementation definition* that contains *service* and *port* elements as shown in the last part of Figure 3-5.

WSDL is extended by augmenting Quality Criteria XML Schema (see Appendix A) in the <service> element as shown in Figure 3-5.

The quality specification offered by the service provider contains the same XML structure, but does not include *weight* child element within each quality sub-criteria and using *Promised* child element within *qValue* element instead of *Preferred* one.

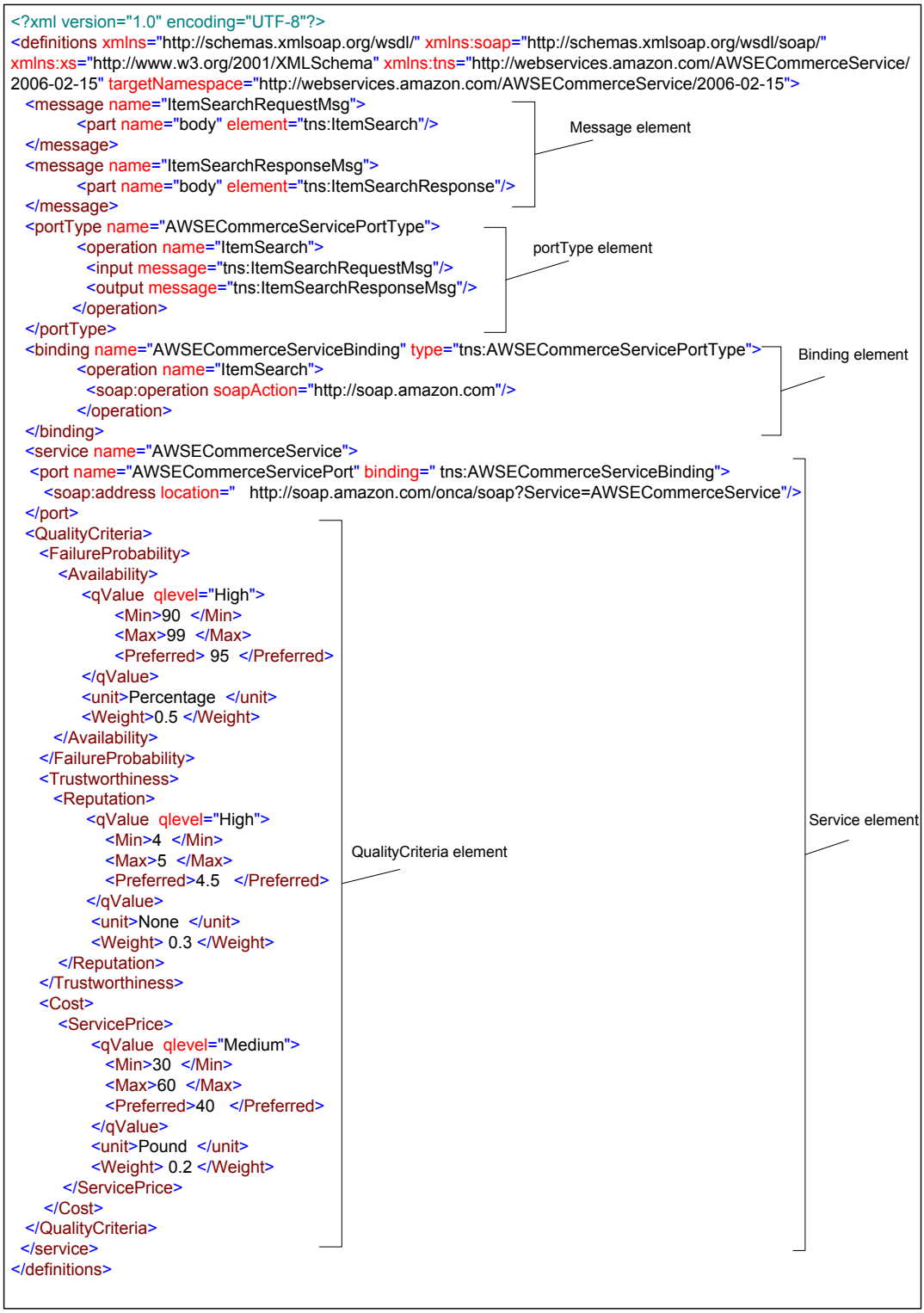

Figure 3-5 Screenshot showing an example of Quality Requirement in Amazon Web Service' WSDL extended with Quality Criteria Classification

3.4.2 Extended UDDI

UDDI provides a registry of businesses and Web services. UDDI describes business by their physical attributes such as name and address and the services they provide. Business services are associated with tModels which can be associated with description standards such as WSDL or taxonomies such as NAICS [68]. The current UDDI allows search to be carried out on limited attributes of services such as *service name*, *key Reference* (which must be unique for a service), or based on a *categoryBag* (which list all the business categories within which a service is listed). Because UDDI does not represent service quality capabilities, it can't search for services on the basis of quality criteria [135].

Various approaches were used in order to enable UDDI to support quality-of-service capabilities. Farkas and Charaf [98] extend UDDI inquiry API with two methods(*find_business_qos* and *find_service_qos*), which correspond to the QoS queries. These methods are used to choose the best available Web service. Ran [5] Extends UDDI data structure with *qualityInformation* data structure under the *businessService* data structure which provide different categories of quality of service information about a particular service, such as availability, reliability, etc. Ali et al. [30] extend UDDI as "UDDIe" which supports the notion of "blue pages". UDDIe enables discovery of services based on QoS attributes by extending the *businessService* class in UDDI with *propertyBag*.

This thesis enables the current UDDI in the proposed quality-based Web service architecture (QWSA) to publish and discover Web services based on the proposed quality criteria classification by extending the current Web services architecture with quality server. Quality server registers quality specifications in its database by using quality manager and enables service discovery and selection based on quality criteria by using quality matchmaker component as is described in Section 4.2.

3.5 Summary

This chapter proposes a quality criteria classification that is required to be augmented within the WSDL to enable the requester to select the best services based on quality issues and to achieve his/her satisfaction. The quality criteria classification consists of four groups: Performance, Failure Probability, Trustworthiness, and Cost. Each group consists of several quality sub-criteria.

The current Web services standards; Web Services Description language (WSDL) and Universal Description Discovery and Integration (UDDI) do not support quality-of-service capabilities. This chapter associated Quality criteria XML Schema (the implementation of the quality criteria classification) within the WSDL. The extension is made in the *Service Implementation Document* part by adding a new element tag called <QualityCriteria>. Also, this chapter enables the current UDDI to publish and discover Web services based on quality criteria by extending the current Web services architecture with quality server.

The quality classification enables the proposed quality-based Web service architecture (QWSA), which is described in Chapter 4, to select the best available services based on quality aspects.

Chapter 4 QWSA: A Proposed Quality-Based Web Service Architecture

4.1 Introduction

Since Web services can be provided by third parties and invoked dynamically over the Internet, their quality can vary greatly. It is important to have a framework capturing the quality of the Web services provided by the provider as well as required by the requester, and the quality matchmaking to explore and select the best Web service.

Section 4.2 introduces a quality-based Web service architecture (QWSA), which extends the IBM Web service architecture with quality server. The quality server acts on behalf of the requester to select the desired Web services. The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The role of each component is introduced.

4.2 The Components of the Quality-Based Web service Architecture

The current Web services architecture does not offer comprehensive quality support. The UDDI is just a registry database and service discovery engine and it allows requesters to look for Web services based on their functionality. UDDI does not represent service quality capabilities that can't search for services on the basis of quality criteria [135].

Different approaches have introduced for enhancing the current Web services architecture to support quality aspects. The current Web services architecture is extended with a QoS broker in [94] [98], [86], [84], [91] in order to select the service. However, the aforementioned QoS brokers are not well defined; they do not describe the details of the service selection process.

This thesis proposes a quality-based Web service architecture (QWSA), which extends the IBM Web service architecture with a quality server [136]. This extension enables the current UDDI to publish and discover Web services based on the proposed quality criteria classification by extending the current Web services architecture with a quality server. The quality server registers quality specifications in its database and enables service discovery and selection based on quality criteria.

The QWSA as shown in Figure 4-1 has four components: service requester, service provider, quality server, and UDDI registry. These components interact with each other to discover and select the desired advertised service.

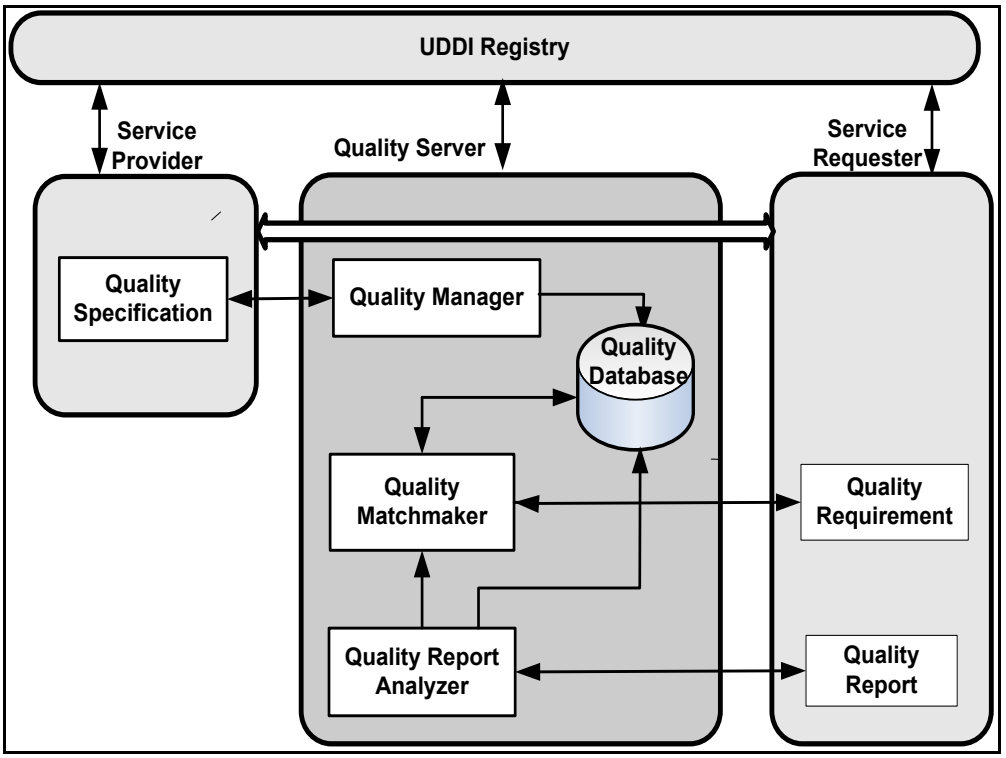

Figure 4-1 Quality-Based Web Service Architecture (QWSA)

These components and their responsibilities are described below.

1) Service Provider

This thesis makes two assumptions on service providers:

- Service providers describe their services based on their functionality and quality specification, and publish the Web services based on their functionality (such as the service name, service access point, UDDI classification of the service, etc.) in the current UDDI registry. Whereas, the service providers send the quality specification of their services to the quality server and store in its database. Service providers separate the service's functionality from quality specification because the current UDDI registry is not designed to accept quality specification and does not allow the requester to look for Web services based on their quality issues.
- Service providers describe their services associated with quality specification using the WSDL standard. WSDL is extended with the quality specification based on the proposed quality criteria classification.

2) Service Requester

Service requester has the following tasks:

- Service requester sends his/her request including both the functional requirements as well as the quality requirements to quality server and let the server select the most suitable Web service on behalf of him/her. If the result is not satisfying the requester, then he/she can reduce his/her quality of service constraints or consider trade-offs between the desired qualities of service [5].
- After invoking the service, requester submits a quality report regarding his/her feeling about the service. The quality report is sent to the quality report analyzer for processing.

3) UDDI Registry

UDDI is a registry that allows the service providers to publish their services and the service requesters to look for Web services based on their functionality but not quality specifications. To enable current UDDI to publish and discover Web services based on quality specifications, the IBM Web service architecture is

extended by quality server. Quality server registers quality specifications provided by service providers in its database by using quality manager and enable service discovery and selection based on quality criteria by using quality matchmaker. The quality server and its components are described below.

4) Quality Server

The quality server is a separate component from requesters and providers. It enables the server to make independent decision and to be independent of the application domain. The quality server consists of four main components:

(1) Quality Manager.

(2) Quality Matchmaker.

(3) Quality Report Analyzer.

(4) Quality Database.

The quality server provides the following tasks:

- Enhance the current UDDI role by enabling service publishing and discovering based on quality criteria
- Quality server collects quality specifications about Web services provided by the service providers. By doing so, it enables the service providers to register their quality descriptions.
- Quality server submits a query to UDDI registry on behalf of the requester for services' functional information such as service name, service URL, service category, etc.
- Quality server holds up-to-date information on quality specifications currently available for services.
- Quality server matches the quality specifications against the quality requirements.

- Quality server makes service selection decisions for requester. The service selection is based on the mathematical model, which uses the Analytical Hierarchy Process (AHP) and the Euclidean distance. So, the quality server assists the requester to choose the best available service based on quality criteria.

There are two types of queries:

Volatile query: The requester sends a query to UDDI, the matched services are immediately returned, and then the query is discarded by the UDDI [70]. This thesis assumes that the query is volatile that is no new services will be added to UDDI and no changes to the quality criteria values for this service.

Persistent query: The requester sends a query to UDDI. This query is persistent as it remains valid for a long time defined by the requester. The matched services are returned. Within the valid period of the query, when the new matched service is added to UDDI or has been changed, the UDDI notifies the requester of the new of matched services. The persistent query is removed when the validity period is ended [70].

Through the quality server, service providers can augment their Web services' specifications with quality criteria while a requester can define its requirements related to quality criteria.

The four quality server components and their functions are described below.

Quality Manager

The quality manager has the following tasks:

- When the service providers publish their Web services with functional descriptions to UDDI registry, the quality manager collects quality specifications of the corresponding published services in the UDDI from the service providers and places them in the quality server's database. The quality specifications are required for quality matchmaking and selection.

- Quality manager stores the services information such as endpoint, URL, and functional name in quality server's database based on their categorization (tModel) by using matchmaking process which is described in Chapter 5.
- Quality manager updates regularly the quality server's database whenever significant changes happen, to keep the server's information consistent and up to date with UDDI registries.
- Quality manager checks regularly the available services for new quality specifications. Once an offer expires, it is deleted from the quality server database.
- Quality manager maintains the quality statistical information generated by the quality report analyzer.

Quality specifications include services with different quality criteria. Table 4-1 shows an example of three service's levels offered by the service providers with different quality criteria values.

Table 4-1 Service Levels with Quality Criteria

Service Levels	High	Medium	Low
Processing Time	2msec	5msec	8msec
Throughput	500 request/s	200 request/s	100 request/s
Availability	99%	80%	60%

Quality Matchmaker

The quality matchmaker is the core of a quality server. Before a requester binds to Web services and begins to execute its tasks, the quality matchmaker must first determine whether the service quality desired by the requester can be achieved.

Quality matchmaker has the following tasks:

- When the requester sends the service request including both the functional requirements and quality requirements to the quality server, a quality matchmaker matches:
  - The functional requirements with the functional specifications in the UDDI registry.
  - The quality requirements with the quality specifications in the quality database, by using quality criteria classification (see Section 3.3) and mathematical model (see Section 5.5).
- Quality matchmaker discovers and selects the best available Web service on behalf of the requester. The Web service selection

Quality matchmaker component is described in details in Chapter 5.

Quality Report Analyzer

The quality report analyzer has the following tasks:

- After the Web service is consumed, the requester sends a quality report based on his judgments on the services to quality report analyzer, which can be subjective.
- The quality report includes information such as service location, invocation date, service execution duration, quality criteria offered, service rank, and comments as shown in Table 4-2.
- The quality report analyzer produces statistical information about the service and store them in the quality server's database as the historical quality information. The statistical information contains the Reputation criterion which depends on the "service rank" that the requester can assign for the service after invoking it. The value of the Reputation is calculated using the equation $q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the requester ranking on a service's

reputation, n is the number of times the service has been graded. The service rank are given to the requesters, for example, the range is between [0,5] [110].

- The quality matchmaker uses this quality information for future service matchmaking and selection.

Table 4-2 Example of Quality Report

Quality Report	
Service URL	http://architag.com/WeatherInfo
Invocation Date	1/9/2004
Sevice Execution Duration	40 msec
Quality Criteria offered	Processing Time, Throughput, Availability
Service Rank	4
Comments	…..

Quality Database

The quality database stores the information retrieved by the quality manager and quality report analyzer. The information stored in quality database includes:

- Service functional specifications retrieved from the UDDI registry, such as service endpoint, URL, function name, description, etc.
- Quality specifications retrieved from the service providers, such as availability, service price, etc.
- Statistical information of each service which produced by quality report analyzer, such as reputation.

The quality manager collects the first two service specifications (functional and non-functional) and stores them in the quality database. The quality report analyzer collects requester's quality report and stores it in the quality database as a historical data.

The quality information stored in quality database will be used by quality matchmaker for selecting the best candidates Web service.

4.3 A case of Using QWSA

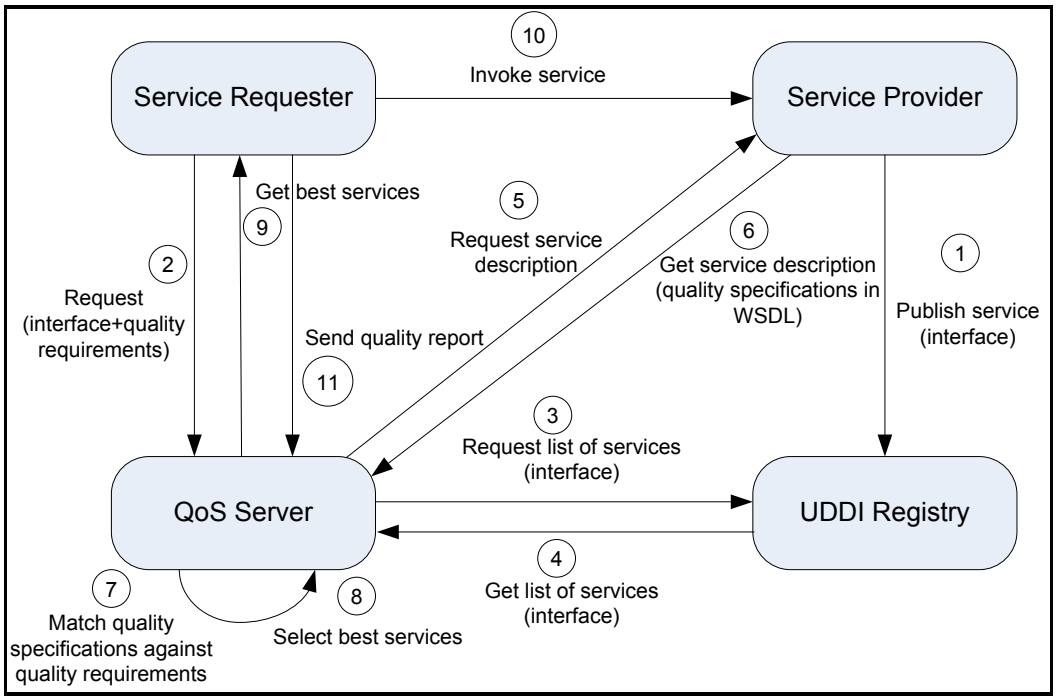

Figure 4-2 Interactions between the four participating roles in QWSA

Figure 4-2 illustrates one possible sequence of interactions between the components of quality-based Web service architecture (QWSA). All communications between the components uses SOAP messages. These interactions are listed below:

1. Service providers register their services in the UDDI registry.
2. Service requester sends quality request to quality server. The requester may use graphical user interface (GUI) (as it is used in this thesis; see Chapter 6) to specify requests include services implementing interfaces and quality requirements associated with weights regarding requester's quality criteria preferences (This process is discussed in details in the implementation chapter; see Chapter 6).
3. When quality server receives an inquiry from the requester, it searches the UDDI registry for related results.

4. Quality server gets a list of services implementing interfaces and stores it in the quality database.
5. Quality server requests service providers for service descriptions augmented with quality specification related to the list of services stored in quality server's database.
6. Quality server gets the result and stores it in the quality database.
7. All the discovered Web services can be ranked between the shortest distance and the farthest distance by using Euclidean distance technique.
8. Then the quality server selects the service with shortened distance as the best available Web service .This step is discussed in section 5.3.
9. Quality server sends a list of best services to service requester.
10. If requester is satisfied with the result, he/she invokes the service, and if the result is not satisfied then the requester can change the request with different quality preferences associated by reducing the quality criteria values or considering trade-offs between the desired qualities of service [137], [5].
11. After the requester invoke the service, he/she sends a quality report to quality server as a feedback and be stored in the database as historical quality information which can be used in the future selection.

4.4 Summary

This chapter proposes quality-based Web service architecture (QWSA). The proposed QWSA extends the IBM Web service architecture with quality server, because the current Web service architecture does not offer comprehensive quality support. The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database. The roles of each component are introduced. The main purpose of the quality server is to assist requester to select the best available Web service based on quality criteria.

Chapter 5 A Theoretical Model of Service Selection

5.1 Introduction

Web services architecture, and standards, support publishing, finding, and binding to services. However, between finding and binding operation, there is another operation, which is service selection wherein a specific service is chosen by a prospective requester. In addition, the number and diversity of Web services grows exponentially, and the Internet is an open environment, where information sources and communication links are unpredictable. With the ever growing number and diversity of Web services, enhanced techniques for service discovery and selection are desperately needed.

This chapter introduces the quality matchmaker as a core component in the quality server of the proposed quality-based Web service architecture (QWSA).

The mathematical model is explained in Section 5.2 and Section 5.3. The quality matchmaker components and their roles are described in Section 5.4, which is the most important stage in the matchmaking algorithm. The mathematical model uses two techniques: Analytical Hierarchy Process (AHP) and Euclidean distance in order to select the best candidate Web services based on requester's quality preferences and quality levels (High, Medium or Low). The quality matchmaking algorithm is illustrated by using an example from the Amazon E-Commerce service (AEC) case study in Section 5.5.

5.2 Modelling Quality Service Selection

Most of the related quality service selection approaches depend on matchmaking mechanisms. The matchmaking mechanism either using semantics approaches as in [76], [70], [103], [33], [108], [38] or computation approaches as in [90], [72].

The quality service selection in this thesis depends on the quality matchmaking process (QMP), which is described in Section 5.4. QMP is based on the mathematical model , which is similar to the QoS matchmaking algorithm that is presented in [72]. The QoS matchmaking algorithm is based on the QoS computation model. The QoS computation model uses the Euclidean distance measure in order to find the nearest Web service to the QoS specifications of the requester that is to find a Web service with a minimum Euclidean distance. The QoS matrix is normalised by using maximizing and minimizing equations that considering the type of the QoS parameter. For example, Response Time needs to be normalized by minimization using the minimizing equation while Availability needs to be normalized by maximization using maximizing equation. But the QoS computation modeldoes not consider the service requester's quality preferences of the quality criteria and therefore does not consider the weight or priority of each quality criteria.

The proposed mathematical model uses two methods in order to select the best Web service. Analytical Hierarchy Process (AHP) method is used to calculate the quality criteria weights based on service requester quality preferences. Euclidean distance method is used as in [72], to measure the distance between the quality requirements specified by the service requester and the quality specifications specified by the service provider. The Web service with minimum Euclidean distance is the best service to select. The mathematical model is described in the following sections.

5.2.1 Problem Definition

This thesis assumes that there is a set of Web services S of *n* available web services with identical functional properties, $S = \{S_1, S_2, ..., S_n\}$. It also assumes that all services are characterized by the same set of *m* quality criteria, $C = \{C_1, C_2, ..., C_m\}$.

The performance of any service in terms of each quality criterion can be measured and represented in a performance matrix $P = \{p_{ij}\}$ of the type:

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \quad [1]$$

Each column of the performance matrix *P* corresponds to a specific web service published by the service providers and each row corresponds to a given offered quality specification criterion, so any element of this matrix p_{ij} represents the performance measure of the *j*-th service S_j in terms of the *i*-th quality criterion C_i.

Requester requirements with respect to all quality criteria are given as a vector of *m* elements, $r = (r_1, r_2, ..., r_m)$, where the element r_i represents the quality required preferred value of service in terms of the *i*-th criterion. The requester's preferences on the importance of all quality criteria should be assessed and represented as a vector of criteria weights $w = \{w_1, w_2, ..., w_m\}$.

The problem is to select a service that best matches requester's quality requirements by considering the weights of quality criteria that based on requester's quality preferences.

5.2.2 Assigning Criteria Weights

Criteria weights could be assigned either directly or indirectly to a service requester. Direct assessment requires a scale, for instance from 1 to 10, where larger scale values represent greater importance of the quality criteria. However, indirect assessment via pair wise comparisons, as shown below, yields more precise criteria weights, which better correspond to requester's preferences.

The method of pair wise comparisons, used in the well-known Analytic Hierarchy Process [138], [139], requires a set of comparison judgments to be provided by the requester. Comparing any two criteria C_i and C_j, the requester assigns a numerical value a_{ij}, which represents the relative importance of preference of quality criterion C_i over C_j. Saaty in [139] suggested a nine-point relative scale measurement as shown in Table 5-1. If the criterion C_i is preferred to C_j, say three times, then a_{ij}=3. If both criteria are equally important, then $a_{ij} = 1$. Obviously, the comparison judgments satisfy the reciprocal property $a_{ji} = 1/a_{ij}$.

A full set of comparisons for *m* criteria requires *m (m-1)/2* judgments. In such a way a positive reciprocal matrix of pair wise comparisons $A = \{a_{ij}\}$ can be constructed:

$$A = \begin{bmatrix} 1 & a_{12} & \cdots & a_{1m} \\ a_{21} & 1 & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & 1 \end{bmatrix} \qquad [2]$$

The criteria weights are calculated from this matrix by the using the following equation [140]:

$$w_i = \frac{1}{number of criteria}\left[\begin{array}{l}\frac{prefernce\ of\ criterion\ i\ \ in\ column\ 1}{sum\ of\ the\ entries\ in\ column\ of\ \ criteria\ 1} + \\ \frac{prefernce\ of\ criterion\ i\ \ in\ column\ 2}{sum\ of\ the\ entries\ in\ column\ of\ \ criteria\ 2} + \\ \frac{prefernce\ of\ criterion\ i\ \ in\ column\ m}{sum\ of\ the\ entries\ in\ column\ of\ \ criteria\ m}\end{array}\right] \quad [3]$$

The set of m relative weights is normalized to sum of one,

$$\sum_{i=1}^{m} w_i = 1,\ w_i > 0,\ i = 1,2,...,m, \quad [4]$$

Therefore the number of independent weights is (m-1).

After constructing the pair-wise comparison matrix and obtaining the criteria weights, the next step is to determine the consistency of the criteria judgements. The Consistency Ratio (*CR*) is used to measure the consistency in the pair-wise comparison matrix A [141]. Matrix A is consistent if the following condition is satisfies [139].

$$a_{jk} = \frac{a_{ik}}{a_{ij}}\ , \text{where } i, j, k=1,\ldots,m. \quad [5]$$

The consistency can be determined by the measure called Consistency Ratio (*CR*), defined as [140]:

$$CR = \frac{CI}{RI} \quad [6]$$

where *CI* is the consistency index and *RI* the random index. The Random Index *RI* value is selected from Table 5-2.

Consistency Index (*CI*) is defined as [142], [13]:

$$CI = \frac{(\lambda_{\max} - m)}{(m-1)} \quad [7]$$

Where $\lambda_{\max}$ is the largest eigenvalue of matrix A, and it is calculated from the following:

1. Calculate the weighted sum matrix by the following [142]:

$$w1\begin{bmatrix} a11 \\ a12 \\ \dots \\ a1n \end{bmatrix} + w2\begin{bmatrix} a21 \\ a22 \\ \dots \\ a23 \end{bmatrix} + \dots + wn\begin{bmatrix} an1 \\ an2 \\ \dots \\ ann \end{bmatrix} = \begin{bmatrix} w_s 1 \\ w_s 2 \\ \dots \\ w_s n \end{bmatrix}$$

2. Divide all the elements of the weighted sum matrices by their respective priority vector element to obtain:

$$\lambda_1 = \frac{w_s 1}{w1} \quad , \lambda_2 = \frac{w_s 2}{w2} \quad , \ldots, \lambda_n = \frac{w_s n}{wn}$$

3. $\lambda_{\max}$ can be obtained from the average of the above values:

$$\lambda_{\max} = \frac{(\lambda_1 + \lambda_2 + \dots + \lambda_n)}{n}$$

If the Consistency Ratio (*CR)* in equation (6) is high, this means that the requester's preferences are not consistent and not reliable. A Consistency Ratio (*CR)* of 0.10 or less is considered acceptable.

Table 5-1 Relative Importance Measurement Scale [139]

Relative Importance Measurement Scale	
Importance Intensity	**Definition**
9	Extremely Preferred
8	Very strongly to extremely
7	Very strongly preferred
6	Strongly to very strongly
5	Strongly preferred
4	Moderately to strongly
3	Moderately preferred
2	Equally to moderately
1	Equally preferred

Table 5-2 Average Random Index (*RI*) [139]

Average random index (RI)										
Size of matrix	1	2	3	4	5	6	7	8	9	10
Random index	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

5.3 Applying the Mathematical Model to Service Selection

This section proposes a method that applies the mathematical model described in Section 5.2 to service selection.

The proposed method is based on the assumption that each criterion has a tendency towards monotonically increasing or decreasing utility, so it is easy to rank all services and locate the best one. Web services should be evaluated on the basis of their closeness to the requester requirements, taking into consideration the relative weights of criteria. In mathematical terms, the closeness between two objects can be expressed by their Euclidean distance ([143] cited [144]), which geometrically is the straight-line distance between two points, representing these objects in the *m*-dimensional space. Therefore, the best service is this one that has the shortest distance from the given requester quality requirements, while the one with the farthest distance is the worst. All other services can be ranked in between these two extremes, with regard to the values of their Euclidean distances.

The proposed method for selecting the best Web service is illustrated with an example as in the following steps:

Step-1: Construct pair-wise comparison matrix

The pair-wise comparison matrix A, equation [2], is constructed with respect to the service requester's quality preferences and compares them in a pair wise way. The pair-wise comparison matrix A is a reciprocal matrix representing the service requester judgements of selecting the relative importance of his preference of quality criterion C_i over C_j from Table 5-1. The main diagonal of the matrix is

always 1. The requester specifies m(m-1)/2 preferences, where m is the number of quality criteria.

Example:

The service requester's quality preferences are:

- Availability (AV) is assigned by the service requester as two times more important than the Reputation (REP).
- Availability (AV) is assigned by the service requester as four times more important than the Price (P).
- Reputation is the same as important as Price.

The number of quality criteria, m=3. The requester specifies 3 preferences or judgments.Thus, a comparison matrix A from the equation [2] is formed:

$$A = \begin{array}{c} \\ AV \\ REP \\ P \end{array} \begin{array}{c} \begin{array}{ccc} AV & REP & P \end{array} \\ \begin{bmatrix} 1 & 2 & 4 \\ 1/2 & 1 & 1 \\ 1/4 & 1 & 1 \end{bmatrix} \end{array}$$

Step-2: Calculate the weight vector of quality criteria

The weights of quality criteria can be calculated from the matrix A by using equation [3].

Example:

$$W(AV) = \frac{1}{3}\left(\frac{1}{1.75} + \frac{2}{4} + \frac{4}{6}\right) = 0.579$$

$$W(REP) = \frac{1}{3}\left(\frac{0.5}{1.75} + \frac{1}{4} + \frac{1}{6}\right) = 0.234$$

$$W(P) = \frac{1}{3}\left(\frac{0.25}{1.75} + \frac{1}{4} + \frac{1}{6}\right) = 0.187$$

The weight vector is:

$$W = [0.579 \quad 0.234 \quad 0.187]$$

Step-3: Calculate the Consistency Ratio (*CR)*

The Consistency Ratio (*CR*) measures the degree of consistency among the pair-wise judgements [145]. It can be calculated from equation [6]. The Consistency Ratio (*CR*) of value 0.10 or less is considered acceptable and the requester judgement is consistent[139]. An acceptable consistency property helps to ensure decision-maker reliability in determining the priorities of a set of quality criteria.

Example:

The Consistency Ratio (*CR)* is calculated from equations [6], [7], and [8] as in the following.

1. Random Index *RI* for matrix A of size 3 is equal to 0.58, as given in Table 5-2.
2. Calculate λ_{max} from the following:

- Calculate the weighted sum matrix by the following:

$$0.579\begin{bmatrix}1\\0.5\\0.25\end{bmatrix} + 0.234\begin{bmatrix}2\\1\\1\end{bmatrix} + 0.187\begin{bmatrix}4\\1\\1\end{bmatrix} = \begin{bmatrix}1.795\\0.711\\0.566\end{bmatrix}$$

- Divide all the elements of the weighted sum matrices by their respective priority vector element to obtain:

$$\frac{1.795}{0.579} = 3.1 \quad , \quad \frac{0.711}{0.234} = 3.04 \quad , \quad \frac{0.566}{0.187} = 3.02$$

- λ_{max} can be obtained from the average of the above values:

$$\lambda_{max} = \frac{(3.1 + 3.04 + 3.02)}{3} = 3.053$$

3. Calculate the Consistency Index CI from equation [7]

$$CI = \frac{(\lambda_{max} - m)}{(m-1)} = \frac{3.053 - 3}{3-1} = 0.0265$$

4. Calculate the Consistency Ratio (*CR)* from equation [6]

$$CR = \frac{CI}{RI} = \frac{0.0265}{0.58} = 0.046$$

The Consistency Ratio (CR) is equal to 0.046 which is less than 0.1, so the pair-wise requester's judgement is consistent and therefore the procedures will continue in order to select the best Web service.

Step-4: Normalize the proposed performance matrix

It is assumed that the performance matrix P is published by the service providers. The service providers publish their Web services with the same functional information but differ with their quality criteria values.

Since the criteria are measured in different measurement units, the performance matrix *P*, equation [1], should be converted into a non-dimensional one. This could be done as each element of *P* is normalized by the following calculation:

$$q_{ij} = \frac{p_{ij}}{\sqrt{\sum_{k=1}^{n} p^{2}{}_{ik}}} \qquad [8]$$

This step produces a normalized performance matrix $Q = \{q_{ij}\}$.

Example:

Suppose that there are three Web services (n=3) have the same functional properties and published by different service providers, characterized by three quality criteria (m=3): C_1=Availability, C_2=Reputation and C_3=Price. The values of the quality criteria are represented in a performance matrix P from the equation [1]:

$$P = \begin{array}{r} AV \\ REP \\ P \end{array} \begin{bmatrix} 95 & 99 & 95 \\ 4 & 3.5 & 3.5 \\ 38.37 & 30.27 & 38.38 \end{bmatrix}$$

The normalized performance matrix can be obtained from equation [8] as shown below:

$$Q = \begin{bmatrix} 0.569 & 0.593 & 0.569 \\ 0.628 & 0.550 & 0.550 \\ 0.617 & 0.487 & 0.618 \end{bmatrix}$$

Step-5: Construct a weighted normalized performance matrix

The normalized values are then assigned weights with respect to their importance to the requester, given by the vector $w = \{w_1, w_2, ..., w_m\}$. When these weights are used in conjunction with the matrix of normalized values $Q = \{q_{ij}\}$, this produces the weighted normalized matrix $V = \{v_{ij}\}$, defined as $V = \{w_i q_{ij}\}$, or

$$V = \begin{bmatrix} w_1 q_{11} & w_1 q_{12} & \cdots & w_1 q_{1n} \\ w_2 q_{21} & w_2 q_{22} & \cdots & w_2 q_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_m q_{m1} & w_m q_{m2} & \cdots & w_m q_{mn} \end{bmatrix} \quad [9]$$

Example:

The weighted normalized performance matrix can be obtained from equation [10]; $V = \{w_i q_{ij}\}$, where w_i is obtained from step-2, as shown below:

$$V = \begin{bmatrix} 0.329 & 0.343 & 0.329 \\ 0.147 & 0.129 & 0.129 \\ 0.115 & 0.091 & 0.116 \end{bmatrix}$$

Step-6: Calculate the relative distances

In this step each of the services is measured according to its closeness to the requester quality requirements. The relative Euclidean distances are calculated as follows:

$$E_j = \sqrt{\sum_{i=1}^{m} (v_{ij} - w_i r_i / \sqrt{\sum_{i=1}^{m} p_{ij}^2})^2} \quad [10]$$

Where *j=1,2,..., n* is the number of Web services.

Example:

Suppose that requester's quality requirements are $r = (98, 3, 40)$ for the corresponding *Availability*, *Reputation* and *Price*. The values of the relative Euclidean distances, measuring the closeness between these requirements and the available services are obtained from equation [11]:

$E_1 = 0.268,\ E_2 = 0.239,\ E_3 = 0.258$

Step-7: Rank services in preference order

This is done by comparison of the values calculated in Step-6. Obviously, the Web service with smallest value $E^* = \min\{E_1, E_2, ..., E_n\}$ gives the closest match to the requester quality requirements and should be selected as the best one.

Example:

It is seen from the result of step-6 that the second Web service is the best one, since its Euclidean distance is smallest (0.239), compared to the distances of other services. So, the requester will select the second Web service.

If the requester's preferences are changed so that the weight vector is:

$W = [W(AV) \quad W(REP) \quad W(P)] = [0.131 \quad 0.677 \quad 0.192]$

Then the Euclidean distance will be:

$E_1 = 0.399,\ E_2 = 0.398,\ E_3 = 0.35$

It is seen that the third Web service is the best for having the smallest Euclidean distance.

This example illustrates that the relative weight given to the quality criteria affects the final ranking of the service and depends on the requester preferences and therefore make certain quality criteria weigh more than others.

In the proposed quality-based Web service architecture (QWSA), it is considered to select more than one best service to be a more efficient approach; if one selected service failed, the others can be used instead.

5.4 Quality Matchmaking

Quality matchmaking is defined as a process that requires the quality matchmaker to match the quality inquiry to all the quality advertisements stored in the quality server's database, in order to find appropriate advertised services, which satisfy the quality requirements specified in the quality inquiry.

Different requesters may have different requirements and preferences regarding quality of Web service. For example, a requester may require to minimize the execution time while satisfying certain constraints in terms of price and reputation, while another requester may give more importance to the price than to the execution time [88]**.** Therefore, a quality matchmaking approach is needed to match quality requirements of requesters with the published quality specifications of providers in order to select the best service based on quality criteria constraints and preferences of the requesters.

The quality matchmaker is the core component in quality server. Every service request received by quality matchmaker will be matched with the service specifications that stored in the quality server database. If the match is successful, the quality matchmaker returns a ranked set of desired Web services and selects the appropriate service based on relevance quality criteria using mathematical technique.

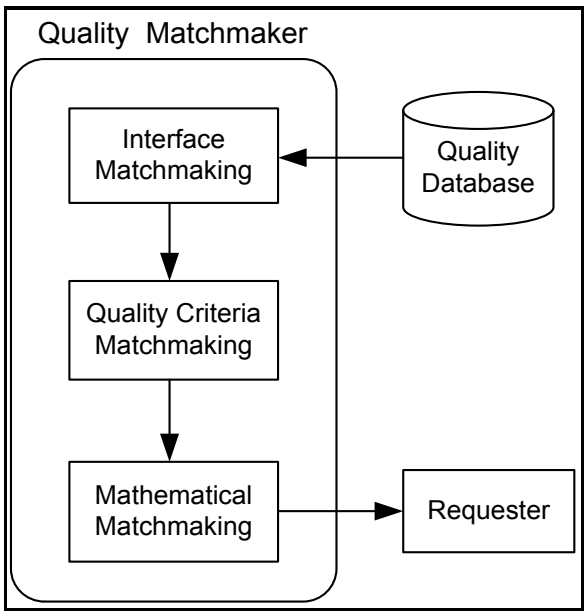

Figure 5-1 Quality Matchmaker

The quality matchmaker component includes the following sub-components (as shown in Figure 5-1)

- Interface matchmaking
- Quality criteria matchmaking
- Mathematical matchmaking

The roles of each sub-component are described in the following:

1) Interface Matchmaking

The interface matchmaking discovers the Web services which fitting functionality with the request requirements. Functionality means an action that either the service or the service requester can do [130]. This step finds all of the services matching the *interface* by using the operation called *find_tModel*() API on the UDDI registry. This step serves as an *interface matchmaking* filter and retrieves a list of all relevant description *tModels* for the services which have the same function. Once a set of tModels that match the specified requirements have been found, then a requester can find the corresponding services by using *find_service*()

operation. This returns a list of all services that implement the description in the chosen *tModel* [71] then quality manager stores the result in the quality database.

The interface matchmaking is important but not sufficient to achieve requester satisfaction, because there are many services implement the same functional properties but have different non-functional (behaviour) properties and need to differentiate between them based on its quality issues. Therefore, further matchmaking is needed regarding quality criteria.

2) Quality Criteria Matchmaking

Quality criteria matchmaking compares quality specifications with quality requirements based on quality descriptions of the services' behaviours. This step reduces or filters the returned list provided by the above interface matchmaking using the *quality criteria matchmaking* filter by considering the structure of the quality criteria XML Schema (as shown in Appendix A). The quality criteria exact match occurs when the group quality criteria type and value (such as Performance, Failure Probability, Trustworthiness, and/or Cost) and its quality sub-criteria type and value (such as Response Time, Availability, Reputation, etc.) are the same for both quality requirements and quality specifications.

Quality criteria matchmaking then uses the *quality value constraint matchmaking* filter in order to reduce the returned last list by satisfying the condition that the value of the required or preferred value of a certain quality sub-criteria type is within the range of the offered quality sub-criteria and also the requested quality sub-criteria range is a subset of offered quality range. Further filtering needed to choose the optimum Web services from this list.

3) Mathematical Matchmaking

Mathematical matchmaking reduces the returned last list of services by using *mathematical matchmaking* filter in order to choose the optimum Web services.

Mathematical matchmaking ranks the services by calculating the distance between the required quality sub-criteria and the offered quality sub-criteria by using the mathematical model. The smallest distance means the best match and therefore the requester can select the best Web services. Once the services are ranked using Euclidean Distance technique, the requester needs to invoke the service by using *find_binding()* operation. This stage is explained in the following section.

5.5 Quality Matchmaking Process

The quality matchmaking process (QMP) determines which Web service from the published Web services is the best service to be selected based on requesters quality requirements and preferences. The matchmaking process is classified into two types:

- The first is the functional (interface) matchmaking that is used to search the UDDI for a Web service with the required functionality.
- The second is to use the quality criteria classification and the mathematical model to match the quality requirements against the quality specifications in the quality database to select the best Web service that fulfils the requester satisfaction and needs.

The quality matchmaking process (QMP) has four algorithms or filters: Interface matchmaking (functional matchmaking), quality criteria type matchmaking (non-functional matchmaking), quality criteria value constraint matchmaking and mathematical matchmaking. Each of these algorithms or filters narrows a set of matchmaking candidates with respect to a given filter criterion. These four algorithms are illustrated below with an example using Amazon E-Commerce Service (ECS) case study (see Appendix D for details).

Step -1: Interface Matchmaking Algorithm:

This step finds all of the matching services that only consider the published Web services matching the required interface. Figure 5-2 shows a flow chart of an

interface matchmaking algorithm that matches the advertised functional specifications in the Web services database with the functional requirements and keeps the result in an *iList* array. This step is evaluated in Section 7.4.1.1.

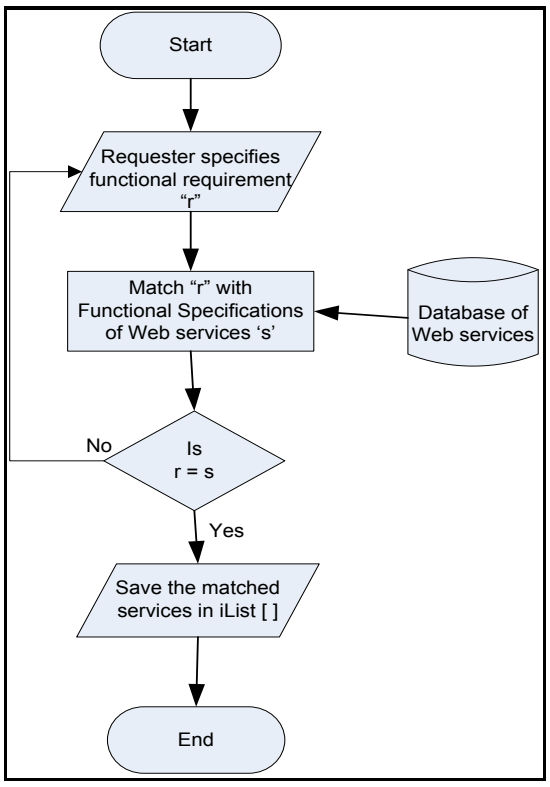

Figure 5-2 Interface Matchmaking Flow Chart

Example:

```
Http://webservices.amazon.com/onca/xml?
Service=AWSECommerceService&SubscriptionId=1NC71HN9R7AE4KJ1G3G2&
Operation=ItemSearch&Title=web services
&SearchIndex=Books&ResponseGroup=Reviews,ItemAttributs,SalesRank,Offers
```

Listing 5- 1 REST Request

The service requester sends his functional requirements to the quality matchmaker. The quality matchmaker sends REST request to the ECS database as

shown in Listing 5- 1. In ECS there are two types of request REST (XML over HTTP) and SOAP request. These request's types are mentioned in Appendix D.

The interface description as shown in Listing 5- 1 includes the following:

- Operation request *ItemSearch*. Amazon E-Commerce Service (ECS) provides two types of inquiries: search and lookup request, see Appendix D.
- SearchIndex *Books*. ECS provides several search indexes: Books, Music, Computer, etc.
- Title *Web Services*. Title is a parameter to the *ItemSearch* operation.
- ResponseGroup: specifies the type of the retrieved information.

The interface matchmaking steps are:

- The quality matchmaker first searches the ECS database using I*temSearch* operation. The matchmaker matches the keyword *Web Services* with the offered books within the *Books* category.
- The matchmaker returns a large list *iList* of matched books includes *Web Services* keyword.

Step-2: Quality Criteria Type Matchmaking Algorithm:

This step is based on quality criteria classification structure. Figure 5-3 shows a flow chart of a quality criteria and sub-criteria matchmaking algorithm. The service requester selects the quality criteria and sub-criteria. The required criteria type (such as Performance, failure Probability, Trustworthiness, and/or Cost) and the sub-criteria type (such as Response Time, Availability, reputation, etc.) are matched with the advertised criteria and sub-criteria type, which are saved in the returned list *iList* in step-1. If both the required and advertised criteria and sub-criteria type are same, then the result is saved in an *sqList*[] array. This thesis for simplicity assumes that the criteria and sub-criteria type of the advertised services are always similar. This step is evaluated in Section 7.4.1.1.

Example:

The above result which stored in *iLis*t is filtered by using quality criteria type matchmaking algorithm. The matchmaker returns a list *sqList* of services contains the following sub-criteria: Availability, Reputation, and Service Price.

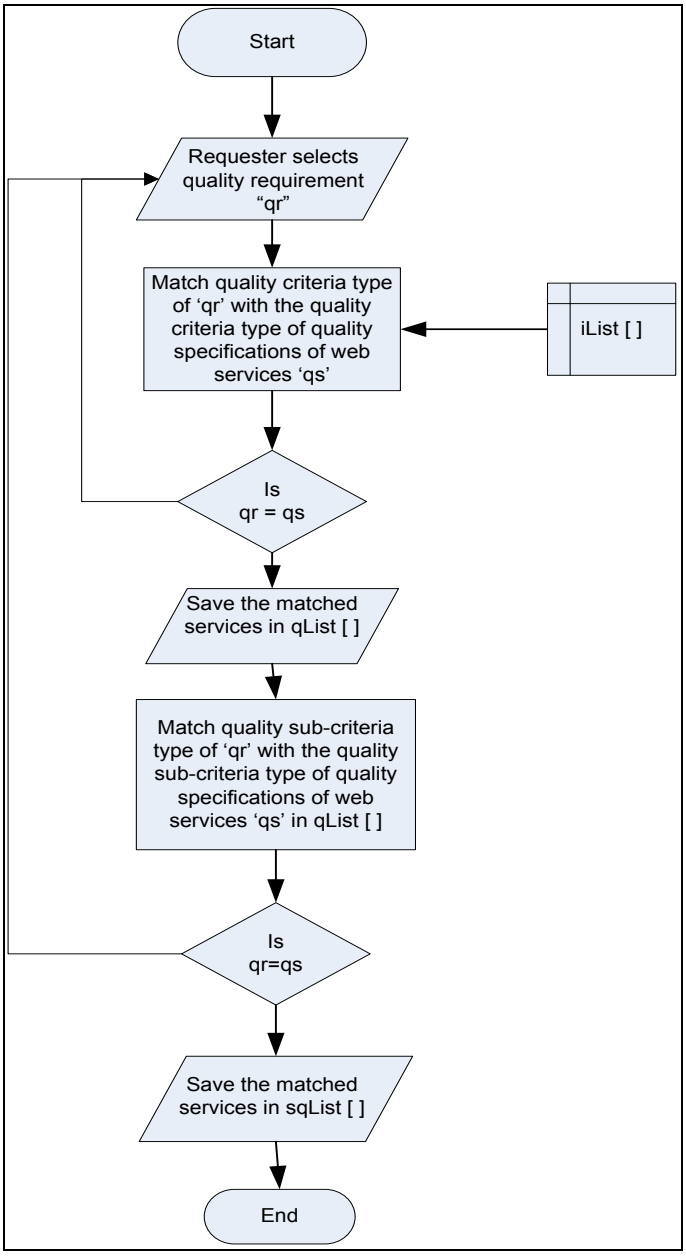

Figure 5-3 Quality Type Matchmaking Flow Chart

Step-3: Quality Criteria Value Matchmaking Algorithm:

This step is based on the quality sub-criteria level (High, Medium, or Low) that the requester specifies. Each quality level has a preferred value. The returned list *sqList* from step-2 is further filtered by using quality criteria value matchmaking algorithm as shown in Figure 5-4. The following rule must be satisfied in order to save the result in *qvList* array list:

qlr<=qls

That is the required quality sub-criteria value must be less than or equal the advertised quality sub-criteria value.

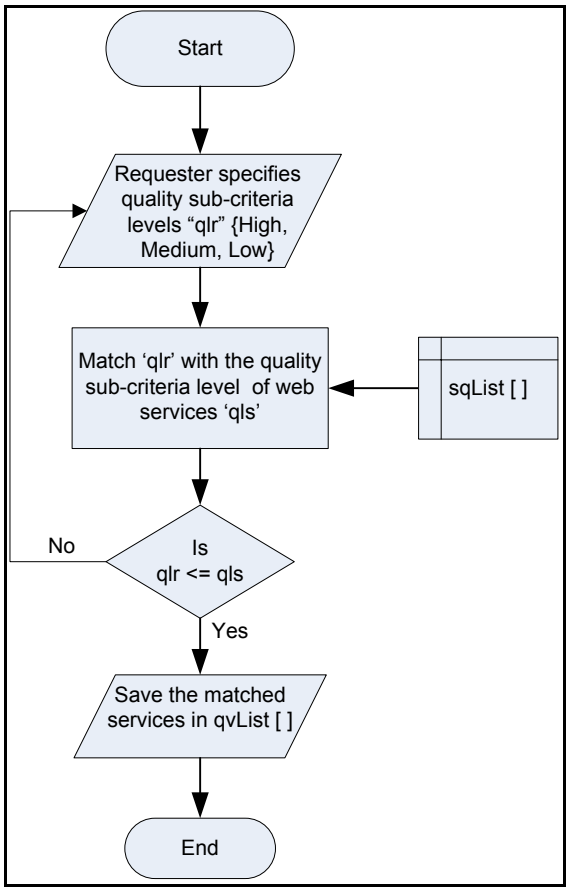

Figure 5-4 Quality Value Matchmaking Flow Chart

Example:

The returned result which stored in *sqList* is further filtered by using quality sub-criteria value constraints matchmaking. The matchmaker returns a list of services *qvList* which their offered quality values are within the range of the required values. The ranges of the required quality values are related to the required quality level parameter *qlevel* (High, Medium, or Low) as shown in Figure. 5-5. The query is shown in Listing 5- 2.

```
SELECT Availability, Reputation, ServicePrice
FROM QualityDatabase
WHERE QualityDatabase.Availability=”High” AND
QualityDatabase.Reputation=”Medium” AND
QualityDatabase.ServicePrice=”Medium”
```

Listing 5- 2 SQL Query

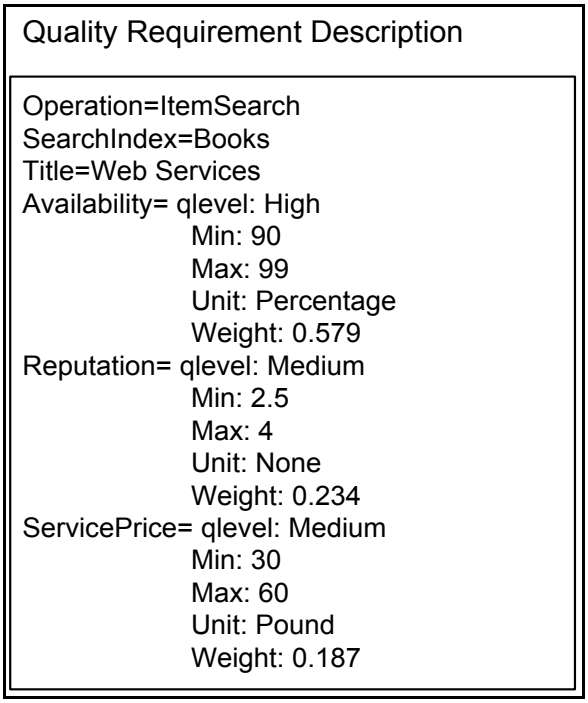

Figure 5-5 Example of Quality Requirement provided by Service Requester

The quality database is the database in the quality server. Figure 5-6 shows the result of quality value matchmaking algorithm. It shows different providers providing services with the same functional specifications but different in its quality specifications.

Quality Specifications Description

Service Provider1	Service Provider2	Service Provider3
Service1 Specification: Title= Understanding Web Service:XML, WSDL, SOAP, and UDDI Availability=98 Reputation=4 ServicePrice=29.07	Service1 Specification: Title= Understanding Web Service:XML, WSDL, SOAP, and UDDI Availability=90 Reputation=4.8 ServicePrice=39.69	Service1 Specification: Title= Understanding Web Service:XML, WSDL, SOAP, and UDDI Availability=99 Reputation=3.5 ServicePrice=30.27
Service2 Specification: Title=Web Services Security Availability=90 Reputation=4 ServicePrice=26.44	Service2 Specification: Title=Web Services Security Availability=95 Reputation=4.8 ServicePrice=42.94	Service2 Specification: Title=Web Services Security Availability=90 Reputation=3.5 ServicePrice=28.47
Service3 Specification: Title=J2EE Web Services Availability=95 Reputation=4 ServicePrice=38.37	Service3 Specification: Title=J2EE Web Services Availability=99 Reputation=4.8 ServicePrice=45.72	Service3 Specification: Title=J2EE Web Services Availability=95 Reputation=3.5 ServicePrice=38.38

Figure 5-6 Example of Quality Specifications Description provided by Service Providers

The result is organised in the following matrix:

$$\begin{matrix} AV \\ REP \\ P \end{matrix}\begin{bmatrix} 95 & 99 & 95 \\ 4 & 3.5 & 3.5 \\ 38.37 & 30.27 & 38.38 \end{bmatrix}$$

The first row is related to sub-criterion Availability (AV), the second row is related to Reputation (REP), the third row is related to Service Price (P).

The first column is related to book with title “J2EE Web Services” which provided by provider 1 (see Figure 5-6), the second column is related to book title “Understanding Web Service: XML, WSDL, SOAP, and UDDI” which provided by provider 3, the third column is related to book title “J2EE Web Services” which provided by provider 3.

Step-4: Mathematical Matchmaking Algorithm

This step is based on the mathematical model that explained in Section 5.2. This step is the most important step in the quality matchmaking process (QMP) and it is implemented in Chapter 6. The mathematical matchmaking algorithm selects the best Web service from the last list *qvList* from step-3 as shown in Figure 5-7. The service requester specifies the selected quality criteria and sub-criteria preferences. The weight of the quality criteria and sub-criteria is calculated using Analytical Hierarchy Process. Then the consistency ratio (CR) must be less than 0.1 to continue the process. Then the Euclidean distance measures the distance between the requester's quality requirements and the provider's quality specifications of the services that are saved in *qvList*[] array from step-3. The service associated with a minimum distance is the best service to select. The AHP and Euclidean distance are explained in Section 5.2.

Example:

The mathematical technique (Analytical Hierarchy process and Euclidean Distance) is used to measure the distance between the quality requirements and the quality specifications. The minimum distance calculated will be the best service to select. After using the mathematical technique the final result are:

The distance of the book title "J2EE Web Services" which provided by provider 1 is: 0.268.

The distance of the book title "Understanding Web Service: XML, WSDL, SOAP, and UDDI" which provided by provider 3 is: 0.239.

The distance of the book title "J2EE Web Services" which provided by provider 3 is: 0.258.

From the above result the minimum distance is 0.239 which is related to the book title "Understanding Web Service: XML, WSDL, SOAP, and UDDI" and provided by provider 3, so this is the best book which the requester can select to

buy. It is noticed from the result that the book with highest Availability value is selected and it is reasonable because the requester specifies the quality level *qlevel* for the Availability sub-criterion to High, whereas for Reputation and Service Price for Medium, this affect to the weight priority of the Availability which is the highest priority (0.579) and therefore affect the book selection.

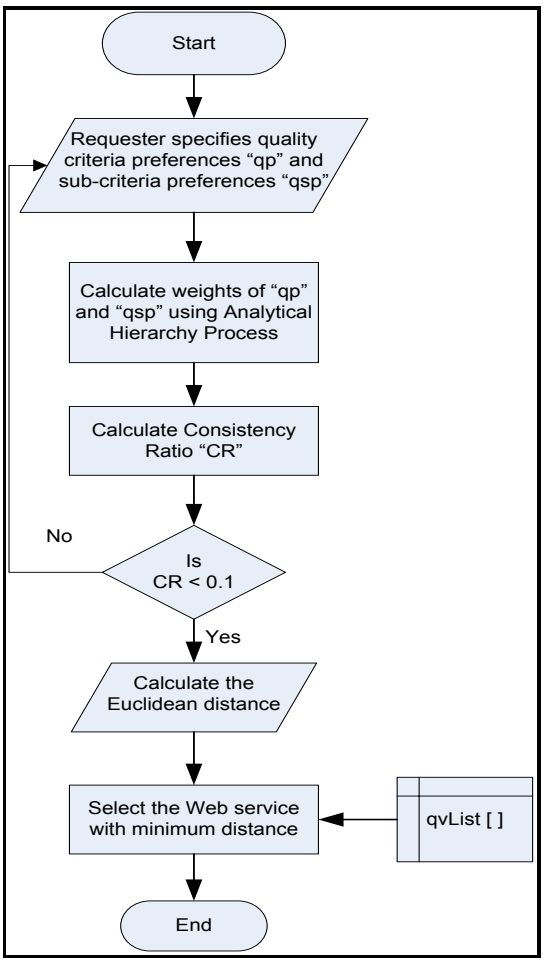

Figure 5-7 Quality Mathematical Matchmaking Flow Chart

5.6 Summary

This chapter has described the role of the quality matchmaker component, which is the core component in the proposed quality-based Web service architecture (QWSA). The quality matchmaker introduces four algorithms or filters: interface matching, quality criteria matchmaking, quality value constraints matching, and

mathematical matchmaking. These four algorithms use the quality matchmaker sub-components to implement their roles. The quality matchmaker ha three sub-components which are: interface matchmaking, quality criteria matchmaking and mathematical matchmaking.

A quality matchmaking process (QMP) is introduced to demonstrate the above four algorithms and to select the best Web service. The last step in the matchmaking process is a mathematical matchmaking algorithm. It is the most important step that uses a mathematical model in order to select the best candidates Web service based on requester's quality requirements and preferences. Two techniques are used in the mathematical model:

1. Analytical Hierarchy Process (AHP) in order to calculate the criteria weights based on requester's preferences.
2. Euclidean distance which measures the distance between the requester's quality requirements and the providers' quality specifications. The Web service with the smallest distance is considered as the best match service to the requester quality requirements.

QMP is illustrated by an example using Amazon E-Commerce Service (AEC) case study. This example shows how the service selection is affected by two factors: the criteria weights and the quality requirements values.

Chapter 6 Implementation of the Quality Matchmaking Process

6.1 Introduction

This chapter presents an implementation of the quality matchmaking process (QMP), which is performed by the quality matchmaker component.

Section 0 introduces a class diagram of the quality service selection system (QSSS). Section 6.3 develops a quality service selection system (QSSS), which is a simulation of the QMP. The QSSS system is a Windows application which enables the service requester to select the best web service based on the quality criteria classification and mathematical model. Section 6.4 presents a sequence diagram of the QSSS system and demonstrates the QMP with an example.

6.2 Designing the Quality Service Selection System

The Visual Studio .NET technology is used to implement the QMP for the following reasons:

- .NET is independence from a specific programming language, which enables the developers to create .NET applications in any .NET-compatible language (Visual Basic, Visual C++ and C#) rather than forcing them to use a single language as using Java language in J2EE.
- Although .NET runs only on a Windows platform, its SOAP capabilities allow components on other platforms to exchange data messages with .NET components, and it is opening up a channel to non-.NET components by integrating XML and SOAP into their messaging scheme [65].

- .NET is developer friendly, easy to use and it is visualised programming. The Framework Class Library (FCL) contains tens of thousands of pre-written classes which are used to create applications [64].
- Visual Studio .NET 2003 development tool is already available in the Lab.

To implement the QMP, Microsoft Visual Studio .NET 2003 software product has been used. Windows Application and C# language (see Appendix H for details) have been used to build a simulation system called "quality service selection system (QSSS). Microsoft Visual Studio .NET 2003 is described in Section 2.4.2.

The QSSS is a user interface which facilitates the service requester to specify the following: his/her quality criteria (Performance, Failure Probability, Trustworthiness and/or Cost) preferences, sub-criteria (Response Time, Availability, Reputation, Service Price, etc.) preferences and the quality sub-criteria requirement values (High, Medium, or Low).

QMP which is described in Section 5.5 is applied in the QSSS with the following assumptions:

- Assume that the QMP occurs in the same domain, for example e-commerce domain as occurred in this thesis.
- Assume that the functional interface matchmaking that matches the advertised functional specification with the functional requirements (step-1 in Section 5.5) is already done and the result of step-1 is stored in the Access database. This assumption is described in Section 7.4.1.1.
- Assume that the returned services in the Access database include the same quality criteria classification (step-2 in Section 5.5), that is, having the same quality criteria (Performance, Failure Probability, Trustworthiness and/or Cost) and sub-criteria (Response Time, Availability, Reputation, Service Price, etc.) types.

Hence, the QSSS supports step-3 (quality value matchmaking algorithm) and step-4 (mathematical matchmaking algorithm) of the QMP (see Section 5.5). This program is described below.

The QSSS consists of class called *Utilities* and window forms. The class diagram in Figure 6-1 shows the relationship between window forms and Utilities class. *Utilities* class contains Matrix class and four methods: *FillMatrix*(), *CalculateWeights*(), *ConsistencyRatio*() and *EuclideanDistance*(). These methods are called by the five window forms: *CriteriaSelection*, *PreferenceSelectio*n, *SubCriteriaSelectio*n, *SubPreferenceSelection* and *RequirementsValue*. The window forms as shown in **Error! Reference source not found.**1 are used to facilitate the requester to specify his/her quality preferences and requirements. *CriteriaSelection* form contains the quality criteria group. *CriteriaSelection* form switches to *SubCriteriaSelection* form if only one criteria group is selected otherwise switches to *PreferenceSelection* form. *SubCriteriaSelection* form contains the quality sub-criteria within the selected criteria group. *PreferenceSelection* form contains the preferences values between the selected criteria group. *SubPreferenceSelection* form contains the preferences values for the selected quality sub-criteria. *RequirementsValue* form contains the quality requirements values for the selected sub-criteria. The form sends a query to the Access database to retrieve list of services associated with matchmaking distance. The service with the minimum distance is the best service to select. The *Utilities* class and each of these Window forms are explained below.

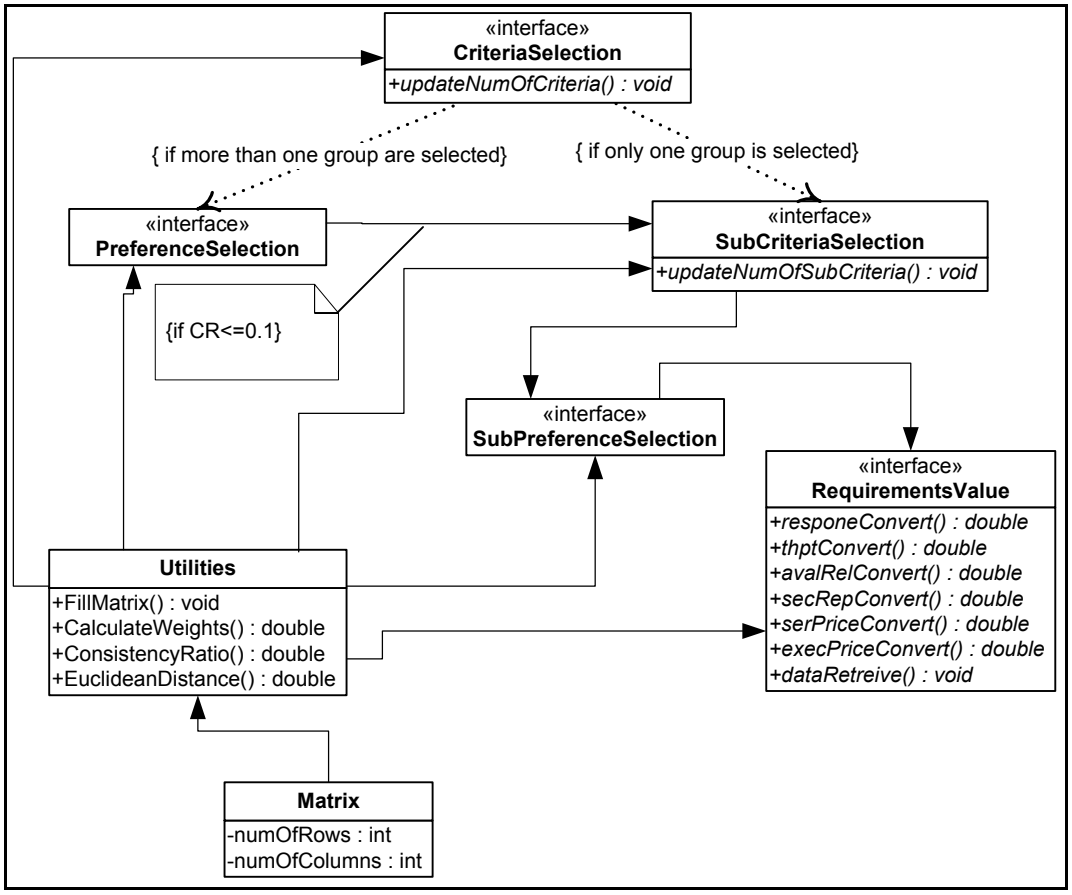

Figure 6-1 Class Diagram of QSSS System

6.3 Implementing the Quality Service Selection System

This section describes an implementation of the quality service selection system (QSSS). In QSSS, there are *Utilities* class and five forms: *CriteriaSelection*, *PreferenceSelection, SubCriteriaSelection, SubPreferenceSelection* and *RequirementsValue.* The functions of the class and each form are explained below.

6.3.1 Utilities Class

Utilities class contains the *Matrix* class and methods such as: *FillMatrix*(), C*alculateWeights*(), *ConsistencyRatio*() and *EuclideanDistance*(). The matrix class and the methods are described below.

Matrix class

Matrix class is used to create matrix instances. The matrix is a multidimensional array as shown in Listing 6-2.

```
public class Matrix
      {
            double[,] matrix;
            int numberOfRows, numberOfColumns;
            public Matrix(int rows, int columns)
            {
                  numberOfRows = rows;
                  numberOfColumns = columns;
                  matrix = new double[rows, columns];
            }

            // Constructor to initialize the data in the matrix
            public double this[int i, int j]
            {
                  set { matrix[i,j] = value; }
                  get { return matrix[i,j]; }
            }

            // Return number of rows in the matrix
            public int Rows
            {
                  get { return numberOfRows; }
            }

            // Return number of columns in the matrix
            public int Columns
            {
                  get { return numberOfColumns; }
            }
      }
```

Listing 6- 1 Matrix Class

FillMatrix() method

FillMatrix() method as shown in Listing 6- **2** is used to construct pair-wise comparison matrix A that based on the service requester's quality preferences.

The input parameters to *fillMatrix*() method are the requester's quality preferences. The output of the *fillMatrix*() method is the pair-wise comparison matrix A.

The number of columns and rows of matrix A is equal to the number of quality criteria (i.e. Trustworthiness) or sub-criteria (i.e. reputation), which are selected by the requester from the *CriteriaSelection* form; that is described below.

```
//fillMatrix0 method construct pair-wise comparison matrix based on the service
     // requester's criteria and sub-criteria preferences

           public  void fillMatrix0(Matrix A, double[] arrValue)
          {
           //if the service requester selects only one quality criteria
                 if(A.Rows==1)
                 {
                       for (int i=0;i<A.Rows;i++)
                       {
                             for(int j=0;j<A.Rows;j++)
                             {
                                   A[i,j]=1;
                                   A[j,i]=1;
                             }
                       }
                 }
  //if the service requester selects more than one quality criteria
                 else if(A.Rows>1)
                 {
                       for (int i=0;i<A.Rows-1;i++)
                       {
                             for(int j=i+1;j<A.Rows;j++)
                             {
                                   double nextVal = getNextValue(arrValue);
                                   if(nextVal != -1)
                                   {
                                        A[i,j]=nextVal;
                                     A[j,i]=1/nextVal;
                                        A[i,i]=1;
                                        A[j,j]=1;
                                   }
                             }
                       }
                 }
          }
```

Listing 6- 2 fillMatrix() Method

CalculateWeights() method

CalculateWeights() method as shown in Listing 6-3 is used to calculate the criteria and sub-criteria weights from the pair-wise comparison matrix A. This method is explained in Section 5.2.2.

The input parameters to *CalculateWeights*() method are the matrix A and the number of selected criteria. The output of the *CalculateWeights*() method is an array contains the weights of the selected quality criteria.

```
// calculateWeights() method calculates the criteria and sub-criteria weights
from pair-wise comparison matrix
public  double[] calculateWeights(Matrix MatrixA, int criteriaNumber)
            {
            //calculate the sum of each column in MatrixA
                  criteriaNumber= MatrixA.Rows;
                  double [] Sum = new double[criteriaNumber];
                  for(int j=0; j<criteriaNumber; j++)
                  {
                        for(int i=0; i<criteriaNumber; i++)
                        {
                              Sum[j]=Sum[j]+MatrixA[i,j];
                        }
                  }

            // create the normalized matrix Normalised
            //by dividing each entry in the matrix by its column sum
                  Matrix Normalised = new Matrix(criteriaNumber,criteriaNumber);
                  for(int j=0; j<criteriaNumber; j++)
                  {
                        for(int i=0; i<criteriaNumber; i++)
                        {
                              Normalised [i,j]=MatrixA[i,j]/Sum[j];
                        }
                  }

            //Calculate the weight of each criteria
            //which is equal to the avarage of its corresponding row
                  double [] WeightCriteria = new double[criteriaNumber];
                  double sumOfRow = 0;
                  for(int i=0; i<criteriaNumber; i++)
                  {
                        for(int j=0; j<criteriaNumber; j++)
                        {
                              sumOfRow=sumOfRow+Normalised[i,j];
                              WeightCriteria[i]=sumOfRow/criteriaNumber;
                        }
                        sumOfRow=0;
                  }
                  return WeightCriteria;
            }
```

Listing 6-3 CalculateWeights() Method

ConsistencyRatio() method

ConsistencyRatio() method as shown in Listing 6-4 is used to calculate Consistency Ratio (*CR*). The CR measures the degree of consistency of the selected preferences values of the quality criteria that considered as a condition

for allowing the service requester to continue the selection procedures or to specify new quality preferences values. This method is explained in Section 5.3.

The input parameters to *ConsistencyRatio*() method are the matrix A, the number of selected criteria and the weights array. The output of the *ConsistencyRatio*() method Consistency Ratio (*CR*) value.

```
//ConsistencyRatio() method calculated the Consistenct Ratio (CR)
            public double ConsistencyRatio (Matrix A, double [] weight, int criteriaNumber)
            {
            double consistencyIndex;
                  double consistencyRatio;
            double randomIndex=1;
                  double sum=0;
                  double weightSum=0;
                  double eigenMax;
                  double [] eigenValue=new double[criteriaNumber];
 // the values of Random Index (RI)for differrent number of criteria selected
 // 3<=RI<=10
                  if (criteriaNumber==3)
                  {
                        randomIndex=0.58;
                  }
                  if (criteriaNumber==4)
                  {
                        randomIndex=0.9;
                  }
                  if (criteriaNumber==5)
                  {
                        randomIndex=1.12;
                  }
                  if (criteriaNumber==6)
                  {
                        randomIndex=1.24;
                  }
                  if (criteriaNumber==7)
                  {
                        randomIndex=1.32;
                  }
                  if (criteriaNumber==8)
                  {
                        randomIndex=1.41;
                  }
                  if (criteriaNumber==9)
                  {
                        randomIndex=1.45;
                  }
                  if (criteriaNumber==10)
                  {
                        randomIndex=1.49;
                  }
      //calculate the eigenvalue max
                  for(int i=0; i<criteriaNumber; i++)
                  {
                        for (int j=0; j<criteriaNumber; j++)
                        {
                              weightSum=weightSum+weight[j]*A[i,j];
                        }
                        eigenValue[i]=weightSum/weight[i];
                        weightSum=0;
                  }
                  for(int k=0; k<criteriaNumber; k++)
                  {
                        sum=sum+eigenValue[k];
                  }
                  eigenMax=sum/criteriaNumber;
   //calculate the Consistency Index (CI)
                  consistencyIndex=(eigenMax-criteriaNumber)/(criteriaNumber-1);
      //calculate the Consistency Ratio (CR)
                  consistencyRatio=consistencyIndex/randomIndex;
                  return consistencyRatio;
            }
```

Listing 6-4 ConsistencyRatio() Method

EuclideanDistance() method

EuclideanDistance() method as shown in Listing 6-5 is used to calculate the Euclidean distance of the advertised Web services. The service with the smallest distance is the best one that the service requester can select it. This method is explained in Section 5.3.

The input parameters to *EuclideanDistance*() method are the performance matrix P that contains the advertised services, the number of selected criteria, the weights array and an array of the quality requirement values. The output of the *EuclideanDistance*() method is an array of the Euclidean distance values for all the advertised services in matrix P.

```
 // EuclideanDistance() method calculates the Euclidean distance for each service in the
performance matrix
            public double[]EuclideanDistance(Matrix P, int subCriteriaNumber, int serviceNumber,
double[] Weight,double []requirement)
            {
                  subCriteriaNumber=P.Rows;
                  serviceNumber=P.Columns;
                  double sum=0;
                  double[] Sqrt=new double[subCriteriaNumber];
                  for(int i=0; i<subCriteriaNumber;i++)
                  {     for(int j=0; j<serviceNumber; j++)
                        {
                              sum=sum+P[i,j]*P[i,j];
                        }
                 Sqrt[i]=Math.Sqrt(sum);
                        sum=0;
                  }
    // calculate the normalized performance matrix
                  Matrix PNormalised = new Matrix(subCriteriaNumber,serviceNumber);
                  for(int i=0; i<subCriteriaNumber; i++)
                  {
                        for(int j=0; j<serviceNumber; j++)
                        {
                              PNormalised [i,j]=P[i,j]/Sqrt[i];
                        }
                  }
     // create V matrix by multiplying weight vector with the normalized performance matrix

                  Matrix V =new Matrix(subCriteriaNumber, serviceNumber);
                  for(int i=0; i<subCriteriaNumber; i++)
                  {
                        for(int j=0; j<serviceNumber;j++)
                        {
                              V[i,j]=Weight[i]*PNormalised[i,j];
                        }
                  }
  //multiply the weight vector with requirement value vector
                  double[] wr=new double[subCriteriaNumber];
                  for(int i=0; i<subCriteriaNumber;i++)
                  {
                        wr[i]=Weight[i]*requirement[i];
                  }
                  double[] SqrtC=new double[serviceNumber];
                  for(int j=0; j<serviceNumber; j++)
                  {
                        for(int i=0; i<subCriteriaNumber; i++)
                        {
                              sum=sum+P[i,j]*P[i,j];
                        }
                  SqrtC[j]=Math.Sqrt(sum);
                        sum=0;
                  }
  //calculate the Euclidean distance
                  double[] EucDistance=new double[serviceNumber];
                  double finalSum=0;
                  for(int j=0; j<serviceNumber; j++)
                  {
                        for(int i=0; i<subCriteriaNumber; i++)
                        {
                   finalSum = finalSum +(V[i,j]-(wr[i]/SqrtC[j]))*(V[i,j]-(wr[i]/SqrtC[j]));
                        }
                  EucDistance[j]=Math.Sqrt(finalSum);
                        finalSum=0;
                  }
              return EucDistance;
            }
```

Listing 6-5 EuclideanDistance() Method

6.3.2 Window Forms

In the quality service selection system (QSSS), there are five window forms: *Criteria Selection*, *Preference Selection*, *Sub-Criteria Selection*, *Sub-Preference Selection* and *Requirements Value*. Each of these window forms are described below.

CriteriaSelection Form

From the *Criteria Selection* form, the service requester selects at least one criterion by click the checkbox next to the criteria group. The *Criteria Selection* form includes four criteria groups: Performance, Failure Probability, Trustworthiness, and Cost. Each of these groups consists of several sub criteria, which will be seen, in *SubCriteriaSelection* form.

This form provides the following functions:

- Counts the number of quality criteria selected by calling *updateNumofCriteria()* method as shown in Listing 6-6. The hierarchy of quality criteria in the *CriteriaSelection* form and the quality sub-criteria in the *SubCriteriaSelection* form is based on the quality criteria classification as described in Section 3.3.

```
// count the number of quality criteria selected
static public int numOfCriteria;
private void updateNumOfCriteria()
 {
    numOfCriteria=0;
    if (checkBox1.Checked) numOfCriteria++;//if Performance is selected
    if (checkBox2.Checked) numOfCriteria++;//if Failure Probability is selected
    if (checkBox3.Checked) numOfCriteria++;//if Trustworthiness is selected
    if (checkBox4.Checked) numOfCriteria++;//if Cost is selected
 }
```

Listing 6-6 updateNumOfCriteria() Method

- If the service requester selects only one quality criterion then this form will:

  - Calculate the criterion weight which is equal to "1" by calling *CalculateWeights( )* method form Utilities class. The criterion weight in this case is always equal "1" because the importance or preference value of one criterion compare to itself is always equal "1".
  - Switch to *SubCriteriaSelection* form and skip *PreferenceSelection* form when clicking *Next* button (see Figure B-1 in Appendix B). This is because the criterion preference value is always equal "1".

- If the service requester selects more than one quality criterion then this form will switch to *PreferenceSelection* form in order to compare between these quality criteria by selecting the preference values

The source code of this form is shown in Appendix B.

PreferenceSelection Form

If the service requester selects more than one quality criteria group the *CriteriaSelection* form are selected then the *PreferenceSelection* form will appear.

For example, if the last three criteria group (see Figure B-1 in Appendix B) (Failure Probability, Trustworthiness and Cost) are selected in *CriteriaSelection* form, then the last three preferences will appear in the *PreferenceSelection* form (see Figure B-2). That means the number of service requester's preferences or judgements which calculated from the equation m (m-1)/2 (see Section 5.3) is equal to 3, where m is the number of selected quality criteria. The preference values are specified by clicking each "comboBox "as shown in Figure B-2. The preference values are divided into three parts:

- The more importance, which includes the values (2, 3, 4, 5, 6, 7, 8, 9).
- The less importance, which includes the values (1/9, 1/8, 1/7, 1/6, 1/5, 1/4, 1/3, 1/2).
- The same importance by selecting the value "1" which is the default value.

The *PreferenceSelection* form provides the following functions:

- Enables the service requester to select his/her quality criteria preferences or importance by clicking the combobox next to each comparison probability as shown in Figure B-2.
- Constructs pair-wise comparison matrix A by calling *FillMatrix0( )* method from *Utilities* class. The *valuesArray* [] is an array contains the preferences values of the selected quality criteria. Comparison matrix A is an instance of the Matrix class and filled with requester's quality preferences. This function is described in Section 5.3.
- Calculates the weight vector of selected quality criteria by calling *CalculateWeights*( ) method from *Utilities* class. The weight calculation is described in Section 5.2.2.
- Calculates the Consistency Ratio (*CR*) by calling *ConsistencyRatio()* method from *Utilities* class. The *ConsistencyRation()* method is called if the number of selected quality criteria is more than two and less than or equal 10. The Consistency Ratio (*CR*) calculation is described in Section 5.3.
- If the Consistency Ratio (*CR*) is less than 0.1, then the requester judgements or preferences are consistent he can continue the selection procedure, otherwise, the requester has to specify new preferences values as shown in **Error! Reference source not found.**.
- When the service requester clicks the Next button, *SubCriteriaSelection* form will appear.

The source code of this form is shown in Appendix B.

SubCriteriaSelection Form

SubCriteriaSelection form sub-criteria within the selected criteria group in the *CriteriaSelection* form..

Performance group consists of: Capacity, Response Time, Latency, Throughput and Execution Time. Failure Probability group consists of: Availability, Reliability, Accessibility, Accuracy and Scalability. Trustworthiness group consists of: Security and Reputation. Cost group consists of: Service Price, Execution Price and Total Price.

The hierarchy of criteria groups and its sub-criteria is based on the quality criteria classification which is described in Section 3.3.

The quality criteria groups and its sub-criteria in this form will be *enabled* (see Figure B-3 in Appendix B) if the service requester selects the criteria group from the first form *CriteriaSelection* form. At least one sub-criterion must be selected in each criteria group. For example, if the first two criteria group are selected in the first form (Performance and Failure Probability), then the above sub-criteria will be enabled: Response Time, Throughput*,* Availability and Reliability. When the requester clicks the *Next* button (see Figure B-3), *SubPreferencSelection* form will appear.

The *SubCriteriaSelection* form provides the following functions:

- Counts the number of sub-criteria selected in each criteria group by calling u*pdateNumOfSubCriteria()* method.
- If the service requester selects only one sub-criteria in each enabled quality criteria group then this form will:
    - Calculate the total weight vector which is equal to the quality criteria weight which calculated in *CriteriaSelection* form if one criterion group is selected, or to the criteria weight calculated in *preferenceSelection* form if more than one criterion is selected. The weight calculation is described in 5.2.2.
    - Switch to *RequirementsValue* form and skip *SubPreferenceSelection* form when clicking *Next* button. This is because the sub-criterion preference value is always equal “1”.

- If the service requester selects more than one quality sub-criterion then this form will switch to *SubPreferenceSelection* form in order to compare between these quality sub-criteria by selecting the preference values

The source code of this form is shown in Appendix B.

SubPreferenceSelection Form

If the sub-criteria selected in *SubCriteriaSelection* form is more than one then the preferences probabilities will appear in *SubPreferenceSelection* form (see Figure B-4 in Appendix B). For example, if the service requester selects Response Time and Throughput within Performance criteria and Availability and Reliability within Failure Probability criteria then the first two importance probabilities will be seen in order to specify their preferences. The preferences values in the "comboBoxes"as shown in Figure B-4 are the same as in the *PreferenceSelection* form. The default value is "1".

The *SubPreferenceSelection* form provides the following functions:

- Enables the service requester to specify preferences values (i.e. 1, 2,…, 9) for the selected sub-criteria by clicking the "comboBoxes" in Figure B-4.
- Constructs a pair-wise comparison matrix for each criteria group. For example, if Performance is selected then the comparison matrix contains the preferences values of Response Time and Throughput sub-criteria.
- Calculates the weight of each criteria group individually. For example, the *TotalWeightPerformance()* method calculates the Performance sub-criteria weight in two cases:
  - If the Performance criteria is only selected by the service requester then the total weight of Performance sub-criteria will be:

    Total Weight= [Performance sub-criteria weight (related to preferences values in *Sub-Preference Selection* form)]*[Performance weight (calculated in *Criteria Selection* form)].

- If the Performance criteria is selected with other criteria groups then the total weight of Performance sub-criteria will be:

  Total Weight= [Performance sub-criteria weight (related to preferences values in *Sub-Preference Selection* form)]* [Performance weight (related to preferences values in *Preference Selection* form)].

- The Consistency Ratio (CR) is not calculated in this form because this thesis assumes for simplicity to specify two sub-criteria in each criteria group, and the Consistency Ratio (CR) calculation need more than two sub-criteria.
- When the requester clicks Next button (see Figure B-4) then *RequirementsValue* form will appear.

The source code of this form is shown in Appendix B.

Requirements Value Form

The *Requirements Value* form (See Figure B-5 in Appendix B) contains the requirement value of the selected sub-criteria from *SubCriteriaSelection* form. Each of the quality requirement value or level has the following options: High, Medium (the default value) or Low.

The *RequirementsValue* form provides the following functions:

- Enables the service requester to specify his/her quality requirement level of the selected sub-criteria quality by clicking the "comboBox" in Figure B-5.
- Converts the requirement levels (High, Medium, or Low) of the quality sub-criteria to values based on the selected sub-criteria type (e.g., Availability, Reputation, etc.) and the service domain (e.g., E-commerce). The methods used in QSSS to convert the sub-criteria requirement levels to values are: *responseConvert(), thptConvert(), avalRelConvert(), secRepConvert(), serPriceConvert()* and *execPriceConvert(). avalRelConvert()* and *secRepConvert()* methods. The requirement level is equivalent to *qlevel*

element which assigned in the quality criteria XML Schema (see Appendix A for details).

- Calls the *dataRetreive()* method when the requester clicks *Submit* button see Figure B-5. The *dataRetreive()* method provides the following tasks:
  - Sends a query request based on the service requester's sub-criteria levels (High, Medium, or Low) to the Access database which called Amazon database. The contents of Amazon database will be described in the coming Chapter 7.
- The *RequirementsValue* form is connected to an Access database using *oleDbConnection1* as shown in **Error! Reference source not found.**. The query result is retrieved and displayed in the DataGrid using *oleDbDataAdapter1* and *dataSet1* with ADO.NET (ActiveXDataObjects.NET). *oleDbConnection1, oleDbDataAdapter1* and *dataSet1* appear below the *RequirementsValue* form see Figure B-5. Further information about ADO.NET and Access database connection are explained in Appendix C.
- Matches the quality requirements specified by the service requester with the quality specification that offered by the service provider
- The matching result is stored in the performance matrix (see Section 5.2) called *criteriaOffered* [,] matrix which contains the services with different quality sub-criteria values.
- The Euclidean distance is calculated for each service by calling *EuclideanDistance()* method from *Utilities* class. The Euclidean distance calculation is explained in Section 5.2.
- The service are ranked from the smallest distance to the largest and displayed in the data grid. The first service with smallest distance is the best service that the service requester can select it.

The source code of this form is shown in Appendix B.

6.4 Sequence Diagram of Using Quality Service Selection System

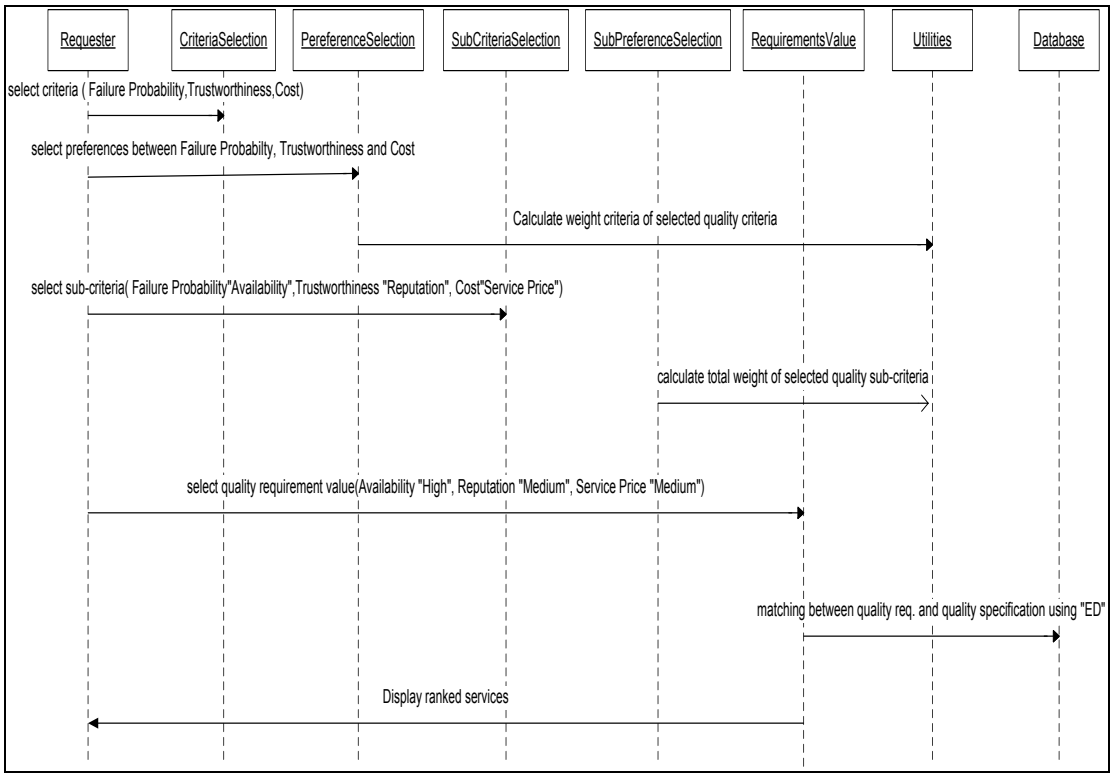

Figure 6-2 Sequence Diagram of Quality Service Selection System

Figure 6-2 shows an example of the quality service selection system (QSSS) process as in the following:

Step-1: Service requester selects the quality criteria; for example, Failure Probability, Trustworthiness and Cos*t* from the *CriteriaSelection*.

Step-2: Service requester specifies the quality preferences between Failure Probability, Trustworthiness and Cost from the *PreferenceSelection* form as the following:

- Failure probability is assigned by the service requester as five times more important than the Trustworthiness.
- Failure probability is assigned by the service requester as two times more important than the Cost.
- Cost is assigned by the service requester as four times more important than the Trustworthiness.

Step-3: Construct pair-wise comparison matrix A by creating an instance of a Matrix class and fill matrix A with the requester's quality preferences by calling *FillMatrix0*() method from Utility class.. The pair-wise comparison matrix A is:

$$A = \begin{array}{c} \\ FP \\ T \\ C \end{array} \begin{array}{c} \begin{array}{ccc} FP & T & C \end{array} \\ \begin{bmatrix} 1 & 5 & 2 \\ 0.2 & 1 & 0.25 \\ 0.5 & 4 & 1 \end{bmatrix} \end{array}$$

Step-4 Call *CalculateWeights*() method from the Utilities class in order to calculate the criteria weight based on requester preferences.

The *CalculateWeight*() method calculates the criteria weights by using the following equation:

$$w_i = \frac{1}{number of criteria}\left[\frac{prefernce\ of\ criterion\ i\ \ in column 1}{sum of the entries in column of\ criteria 1} + \frac{prefernce\ of\ criterion\ i\ \ in column 2}{sum of the entries in column of\ criteria 2} + \frac{prefernce\ of\ criterion\ i\ \ in column\ m}{sum of the entries in column of\ criteria\ m}\right]$$

$W(FP) = \frac{1}{3}\left(\frac{1}{1.7} + \frac{5}{10} + \frac{2}{3.25}\right) = 0.568$; the Failure Probability weight.

$W(T) = \frac{1}{3}\left(\frac{0.2}{1.7} + \frac{1}{10} + \frac{0.25}{3.25}\right) = 0.098$; the Trustworthiness weight.

$W(C) = \frac{1}{3}\left(\frac{0.5}{1.7} + \frac{4}{10} + \frac{1}{3.25}\right) = 0.334$; the Cost weight.

The weight vector is: $W = [0.568 \quad 0.098 \quad 0.334]$

The total weight is equal to 1:

$W(FP) + W(T) + W(C) = 1$

Step-5: Calculate the Consistency Ratio (*CR)*. CR measures the degree of consistency of the selected preferences values of the quality criteria. CR is also calculated if the number of selected quality criteria are more than 2 and less than 10, by calling *ConsistencyRatio*( ) method from Utilities class. If CR value is less than 0.1, then the requester can continue in the selection process otherwise he/she has to specify new quality criteria preferences from *PreferenceSelection* form.

The *ConsistencyRatio*( ) method is calculated by the following:

1. Random Index *RI* for matrix A of size 3 is equal to 0.58, as given in Table 5-2

2. Calculate λ_{max} from equation $Aw = \lambda_{max} w$:

- Calculate the weighted sum matrix by the following:

$$0.568\begin{bmatrix}1\\0.2\\0.5\end{bmatrix} + 0.098\begin{bmatrix}5\\1\\4\end{bmatrix} + 0.334\begin{bmatrix}2\\0.25\\1\end{bmatrix} = \begin{bmatrix}1.726\\0.295\\1.010\end{bmatrix}$$

- Divide all the elements of the weighted sum matrices by their respective priority vector element to obtain:

$$\frac{1.726}{0.568} = 3.04 \quad , \quad \frac{0.295}{0.098} = 3.01 \quad , \quad \frac{1.01}{0.334} = 3.02$$

- λ_{max} can be obtained from the average of the above values:

$$\lambda_{\max} = \frac{(3.04 + 3.01 + 3.02)}{3} = 3.023$$

3. Calculate the Consistency Index CI

$$CI = \frac{(\lambda_{\max} - m)}{(m-1)} = \frac{3.023 - 3}{3 - 1} = 0.0115$$

4. Calculate the Consistency Ratio (*CR)*

$$CR = \frac{CI}{RI} = \frac{0.0115}{0.58} = 0.02$$

CR is equal to 0.02 which is less than 0.1, so the pair-wise requester's judgement is consistent and therefore the procedures will continue in order to select the best book.

Step-6: Service requester selects sub-criteria within each selected quality criteria group from *SubCriteriaSelection* form. For example, the requester selects Availability within Failure Probability criteria group, Reputation within Trustworthiness criteria group and Service Price within Cost criteria group.

Step-7: Calculate the total weight of the selected sub-criteria which equal to the weight of criteria group multiplied by the weight of sub-criteria within the corresponding criteria group, by the following:

Total weight= (criteria weight) * (sub-criteria weight)

Because the requester selects only one sub-criterion in each quality criteria group, the total weight of each sub-criterion is equal to the weight of its criteria group. So, the weight of the Availability (AV) sub-criteria is equal to the weight of Failure probability and equal 0.568, the weight of the Reputation (REP) sub-criteria is equal to the weight of Trustworthiness (T) and equal 0.098, the weight of the Service Price (SP) sub-criteria is equal to the weight of Cost (C) and equal 0.334.

Step-8: The service requester selects the quality requirement levels for each sub-criterion from *RequirementsValue* form. For example, the requirement value for Availability is “High”, Reputaion is “Medium”and Service Price is “Medium”.

Step-9: By clicking the “Submit” button in the *RequirementsValue* form, a query request is sent to the Access database. The result is stored in the Performance matrix.

The performance matrix is retrieved by sending an SQL query to an MS-Access database which contains information about books as shown in Appendix G. The SQL query consists of requirement values (High, Medium or Low) of the selected sub-criteria (Availability, Reputation and Service Price) as shown in Listing 6- **7**.

```
// if Availability's level is High, Reputaion's level is Medium and Service Price's level is Medium
if(boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.Equals("Medium")
&&boxArray[6].SelectedItem.Equals("Medium"))
{
    oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
    ProductTitle,ProductAvailability,SellerReputation,Price FROM AmazonTable WHERE
   AmazonTable.ProductAvailability BETWEEN 80 AND 100 AND
   AmazonTable.SellerReputation BETWEEN 2.5 AND 4 AND AmazonTable.Price BETWEEN
   30 AND 60";
 }
```

Listing 6- 7 SQL Query to an MS-Access database

The requester selects the requirement level of sub-criteria Availability is “High”, Reputation is “Medium”, and Service Price is “Medium” as shown in Listing 6- **7**. The range of requirement values depends on the service domain and on the type of the quality sub-criteria. For example, the requirement level of “High” for Availability is between [80-100], the requirement level “Medium” for Reputation is between [2.5-3.9] and the requirement level “Medium” for Service Price is between [25-49.99].

The query result is saved first in *dataSet1*, using *FillMatrix()* method of *OleDbDataAdapter1*, which is an instance of *OleDbDataAdapter* class that

represents a bridge between a dataset and an OLE DB database. The dataset acts as a local repository of the retrieved data. The data result is then stored in *dataTable*, which is an instance of *DataTable* class and its represents a table of data. The datasets are made up of collections of data tables [64].

The result is then organized in the performance matrix which called *criteriaOffered* matrix. The *criteriaOffered* matrix is an instance of the Matrix class, with rows that contain the sub-criteria fields (Availability, Seller Reputation and Price) and columns that contain the books records as shown in Table 6-1.

Table 6-1 SQL Query Result Obtained for Performance Matrix

Product Name	Seller Name	Availability	Price	Seller Reputation
Service-Oriented Architecture	allnewbooks	80	29.19	2.6
Web Services Platform Architecture	a1books_nj	95	34.11	3.6
Web Services Platform Architecture	alphacraze	82	34.34	3.4
Web Services Platform Architecture	alphacrazeoutlet	97	34.34	3.7
J2EE Web Services	thebookrackrh	80	35.4	2.8
J2EE Web Services	allnewbooks	95	35.49	2.6
J2EE Web Services	alphacraze	99	37.93	3.4
J2EE Web Services	a1books_nj	96	38.51	3.6
How to Break Web Software	powells_books	90	34.99	2.8
Core Security Patterns	fun-for-all58	98	33.85	3.5
Core Security Patterns	allnewbooks	85	38.64	2.6
Core Security Patterns	thebookrackrh	97	39.2	2.8
Core Security Patterns	amz_book	84	39.95	3
Building Web Services with Java	allnewbooks	82	32.34	2.6
Building Web Services with Java	alphacraze	97	34.34	3.4
Understanding SOA with Web Services	superbookdeals	96	25.04	2.5
Understanding SOA with Web Services	amz_book	95	25.95	3
Understanding SOA with Web Services	a1books_nj	86	27.28	3.6
Understanding SOA with Web Services	lphacrazeoutlet	98	27.39	3.7

So, the performance matrix *criteriaOffered* will be created from the SQL query result as the following:

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19
AV	80	95	82	97	80	95	99	96	90	98	85	97	84	82	97	96	95	86	98
P=REP	2.6	3.6	3.4	3.7	2.8	2.6	3.4	3.6	2.8	3.5	2.6	2.8	3	2.6	3.4	2.5	3	3.6	3.7
P	29.19	34.11	34.34	34.34	35.4	35.49	37.93	38.51	34.99	33.85	38.64	39.2	39.95	32.34	34.34	25.04	25.95	27.28	27.39

Step-10: Call *EuclideanDistance*() method from the Utilities class, in order to calculate the distance or the gap between the quality requirement value and the quality specifications which stored in the Performance matrix.

The *EuclideanDistance*( ) method is calculated as in the following:

1. Normalize the performance matrix using the following equation:

$$q_{ij} = \frac{p_{ij}}{\sqrt{\sum_{k=1}^{n} p^2{}_{ik}}}$$

This step produces a normalized performance matrix $Q = \{q_{ij}\}$ as shown below:

	AV	1.922	2.282	1.97	2.33	1.922	2.282	2.378	2.386	2.162	2.354	2.042	2.33	2.018	1.97	2.33	2.386	2.282	2.066	2.354
Q=	REP	0.338	0.468	0.442	0.481	0.368	0.338	0.442	0.468	0.364	0.455	0.338	0.364	0.39	0.338	0.442	0.325	0.39	0.468	0.481
	P	1.156	1.35	1.359	1.359	1.4	1.4	1.5	1.524	1.385	1.34	1.529	1.552	1.582	1.278	1.359	0.991	1.027	1.08	1.084

2. Construct a weighted normalized performance matrix by multiplying the weight vector which obtained from Step-2 with the normalized performance matrix using equation $V = \{w_i q_{ij}\}$. The V matrix will be:

	AV	1.092	1.296	1.118	1.323	1.092	1.296	1.351	1.355	1.228	1.337	1.159	1.323	1.146	1.118	1.323	1.355	1.296	1.173	1.337
V=	REP	0.033	0.046	0.043	0.047	0.036	0.033	0.043	0.046	0.036	0.044	0.033	0.036	0.038	0.033	0.043	0.032	0.038	0.046	0.047
	P	0.386	0.451	0.454	0.454	0.468	0.468	0.501	0.509	0.462	0.448	0.511	0.518	0.528	0.428	0.454	0.331	0.343	0.361	0.362

3. Calculate the relative Euclidean distances using the following equation:

$$E_j = \sqrt{\sum_{i=1}^{m} (v_{ij} - w_i r_i / \sqrt{\sum_{i=1}^{m} p_{ij}^2})^2}$$

Where *j=1,2,…, n* is the number of books in the performance matrix which is equal to 19 and *r* is the requirement values for the sub-criteria as shown below:

$$r = \begin{matrix} AV \\ REP \\ P \end{matrix} \begin{bmatrix} High \\ Medium \\ Medium \end{bmatrix} = \begin{bmatrix} 98 \\ 3 \\ 40 \end{bmatrix}$$

The requirement value for the Availability is "High" which is equal 98 and located within its range [80-100]. The requirement value for Reputation is "Medium" which is equal 3 and located within its range [2.5-3.9]. The requirement value for Service Price is "Medium" which is equal 40 and located within its range [25-49.99]. The aforementioned values for the requirement value: High, Medium and low, can be determined by the system developer or the system administrator and depend on the service domain and on the type of the sub-criteria. For example, the requirement value for buying a book is different than buying a computer and the requirement value of Availability is different than Reputation.

Step-11: Display the services ranked from the smallest distance to the largest distance. The service with the smallest distance is the best one the service requester can select it.

Table 6-2 shows the output result which is based on requester's preferences and it is ranked from the smallest matching distance to the largest one. The matching distance values in Table 6-2 are the values of the relative Euclidean distances, which measuring the closeness between the quality requirements that specified by the service requester and the quality specification that specified by the service providers.

From the output result as shown in Table 6-2, the first book with title "J2EE Web Services" and its provider is "alphacraze" is the best book to select because its matching distance is the smallest (0.387). It is reasonable that the first book is the best because it has the highest Availability value (99%) as seen in Table 6-2 and the required Availability has the highest priority (0.568).

Table 6-2 Output Result

Product name	Seller Name	Matching distance	Seller URL
J2EE Web Services	alphacraze	0.387	http://www.amazon.com/seller=A3H8H6KI3KCVA5
Core Security Patterns	thebookrackrh	0.399	http://www.amazon.com/seller=A1MD3EN9VM2K1F
Core Security Patterns	fun-for-all58	0.4	http://www.amazon.com/seller=A1MD3EN9VM2K1F
J2EE Web Services	a1books_nj	0.404	http://www.amazon.com/seller=A3H8H6KI3KCVA5
Web Services Platform Architecture	alphacrazeoutlet	0.405	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
Building Web Services with Java	alphacraze	0.406	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
Understanding SOA with Web Services	lphacrazeoutlet	0.409	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
Web Services Platform Architecture	a1books_nj	0.419	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
J2EE Web Services	allnewbooks	0.419	http://www.amazon.com/seller=A3H8H6KI3KCVA5
Understanding SOA with Web Services	superbookdeals	0.431	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
Understanding SOA with Web Service	amz_book	0.434	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
How to Break Web Software	powells_books	0.454	http://www.amazon.com/seller=A1C3QU77DDT2KW
Core Security Patterns	allnewbooks	0.482	http://www.amazon.com/seller=A1MD3EN9VM2K1F
Core Security Patterns	amz_book	0.484	http://www.amazon.com/seller=A1MD3EN9VM2K1F
Understanding SOA with Web Services	a1books_nj	0.498	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
Web Services Platform Architecture	alphacraze	0.514	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
Building Web Services with Java	allnewbooks	0.522	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
J2EE Web Services	thebookrackrh	0.529	http://www.amazon.com/seller=A3H8H6KI3KCVA5
Service-Oriented Architecture	allnewbooks	0.548	http://www.amazon.com/seller=A1RAFT0AR298LX

6.5 Summary

The quality matchmaking process (QMP) has been implemented by developing a simulation system called quality service selection system (QSSS). QSSS is developed using Windows application within Visual Studio .NET 2003 tool. QSSS is a user interface, which enables service requester to specify his/her quality preferences and requirements. QSSS consists of the following forms and classes:

- Criteria Selection form
- Preference Selection form
- Sub-Criteria Selection form
- Sub-Preference Selection form

- Requirements Value form
- Utilities class

The functions of each form are described. The Utilities class consists of methods which used to calculate the criteria and sub-criteria weight and to calculate the Euclidean distance between the quality requirement values specified by the service requester and the quality specifications offered by service providers.

A sequence diagram of QSSS system is presented to demonstrate the quality service selection process with an example.

Chapter 7 Evaluation

7.1 Introduction

This chapter evaluates (1) the proposed quality-based Web service architecture (QWSA), (2) the quality matchmaking process (QMP) and (3) the quality service selection system (QSSS). The QWSA is evaluated in Section **Error! Reference source not found.** by comparing it with the related architectures regarding five criteria: scalability, extensibility, commodity to standards, ease of implementation and technique for selection. The QMP is evaluated in Section 7.3 by comparing it with the related matchmaking techniques. The QSSS is evaluated in Section 7.4 through a case study. The efficiency of QSSS and the QMP are discussed in Section 7.5

7.2 Evaluation of the Quality-Based Web Service Architecture

The proposed quality-based Web service architecture (QWSA) is evaluated by comparing it with the related Web services architectures regarding the following five criteria:

1. **Scalability:** It is the capability of a system to increase total throughput and transactions under an increased load when resources or hardware are added.
2. **Extensibility:** It is the ability to extend a system through the addition of new functionality or through modification of existing functionality.
3. **Conformity to standards:** Extending either the Web services' core standards or other higher standards with quality aspects.
4. **Ease of implementation:** The ability to implement the system in an easy way.

5. **Techniques for selection:** Specify the type of the selection technique.

7.2.1 QoS-Capable Web Service Architecture

A QoS-capable Web service architecture (QCWS) which is presented in [94], [92] has three components: server (provider), QoS broker and client. The server assigns different amount of system resources to different clients according to their QoS requirements. The server contains QoS information and QoS admission. The QoS information includes service levels with corresponding costs and maximum service capacities. After a broker selects a service, QoS admission sends request to the server for confirmation (admission). QoS broker acts as a mediator between service providers and clients. It receives clients QoS requests and identifies qualified services for them. Its main components are: QoS information manager, QoS negotiation manager and QoS analyzer. QoS information manager collects QoS information from the server for QoS negotiation. It checks UDDI registry periodically to get up-to-date servers information and contacts servers for QoS information. The collected information is placed in he broker's database. QoS negotiator manager is the core of a QoS broker. It manages service selection for clients. After receiving client's request, it searches through broker's database to look for qualified services. A decision algorithm is used to select the most suitable one. Once the candidate is selected, the QoS negotiation manager negotiates with the server to meet the QoS requirements. If the negotiation is not successful, the broker must identify another candidate server and repeat the negotiation process. QoS analyzer produces statistical information about the server and put them in the broker's database. Clients send their QoS requirements to a broker and let it choose the most suitable server for them.

The evaluation criteria of the QCWS system:

1. Scalability: QCWS supports scalability by providing a QoS Admission and Enforcement component. The admission control compares the number of accepted users with the maximum capacity of the system. If current used

capacity is less than maximum capacity, the server accepts the user's request; otherwise the request is rejected.

2. Extensibility: no information about it.
3. Commodity to standards: no information about it.
4. Ease of implementation: the implementation is easy but not completed; it considers only the number of accepted requesters within the maximum capacity of the system and the number of rejected requesters when the system is overloaded. But not consider how the system selects the service.
5. Technique for selection: The negotiation technique is used to select the best service.

7.2.2 UDDI eXtension Architecture

A UDDI eXtension (UX) architecture is proposed in [95]. UX architecture facilitates requesters to discover services with good qualities. It is comprised of service requester, local UDDI registry, test host and UX server. The service requester queries the UX server to find the matching services and invoke the services. The requester then sends a QoS report about the performance of the service to UX server. The local UDDI registry records the local service description and connected to the UX server as a backend registry. The test host generates QoS reports for the services registered in local registry. The UX server plays an important role in the system. When it receives an inquiry from the requester, it searches the UDDI for related results. The server then sorts the service results according to QoS requirements and sends the result back to the requester. The UX server also receives the requester's QoS report and stores it in a database.

The evaluation criteria of the UX system are:

1. Scalability: UX system supports scalability by using the federated discovery approach. The system be able to scale to support a huge number of requesters and services while adapting the underlying domains' changes.
2. Extensibility: A lookup interface between servers is extended to support the federated discovery. It contains query ID, sender, query response and QoS summery.
3. Commodity to standards: The extended inquiry interface conforms to the UDDI specification.
4. Ease of implementation: the implementation is not completed; the federated discovery hasn't been fully implemented.
5. Technique for selection: Keyword matching in addition to requester's preferences on the service's QoS metrics.

7.2.3 Web Service Quality Broker Architecture

The Web Service Quality Broker Architecture, which is proposed in [86], helps the service requester to find the optimal Web service. The quality broker is located between the requesters and providers. It monitors quality attribute values of registered services and store them in WSLA (Web Service Level Agreement) document. The quality broker performs the negotiation through investigating WSLA details with same function and quality attributes of requester.

The aforementioned related architecture used the WSLA to accommodate the quality attributes in order to be used in the negotiation process to select the optimal Web services. Wherein the proposed QWSA, the WSDL (Web Services Description Language) is extended with quality criteria, which is used further in the quality matchmaking process (QMP) to select the best Web service. Also, the related architecture didn't manage the dynamic nature of the quality criteria that is to keep up-to-date information on quality specifications currently available for

services. But the propose quality server address this issue by its component the quality manager.

The evaluation criteria of the Web Service Quality Broker Architecture are:

1. Scalability: The architecture does not support scalability.
2. Extensibility: The quality model, which is based on the architecture, is extensible that can add more QoS attributes within each Web service quality aspects: Performance, Safety and Cost.
3. Commodity to standards: The WSLA is used to accommodate the quality attributes rather than using the Web service standard description language WSDL.
4. Ease of implementation: No information about the system implementation..
5. Technique for selection: Negotiation process is used to select the best service.

7.2.4 QoS Certifier

A Web service discovery architecture which is proposed in [5] extends the current Web service architecture with Web service QoS certifier in order to discover Web services by considering the functional and non-functional requirements. There are four roles in the proposed architecture: Web service supplier, Web service consumer, Web service QoS certifier and the new UDDI registry. The Web service provider sends its QoS claim to the Web service QoS certifier. The QoS certifier certifies the claim and sends the certification identification information back to the provider. After the QoS certification been issued, the provider then registers the service with both functional description and its associated certified quality in the new UDDI. The new UDDI differs from the current UDDI by having information about the functional description of the Web service as well as its associated certified quality of service information. The consumer searches the new UDDI registry for a service with certain functional and quality of service requirements. Once a Web service result is found, the WSDL and the certified

QoS information are retrieved by the consumer then he/she can invoke the Web service.

The evaluation criteria of the QoS Certifier are:

1. Scalability: The architecture does not support scalability.
2. Extensibility: The architecture extends the current UDDI data structure with *qualityInformation* data structure.
3. Commodity to standards: The architecture is commodity to Web services core standards such as UDDI and WSDL.
4. Ease of implementation: No information about the system implementation.
5. Technique for selection: No information about the selection technique.

7.2.5 Web Service QoS Architecture

A Web service QoS (WS-QoS) architecture in [84], [91], extends the current Web service architecture with Web service broker (WSB) in order to select the appropriate service based on QoS requirements regarding server and network performance. The Web service client contacts the WSB for looking up a service instead of searching the UDDI registry. The WSB checks regularly the UDDI for new offers to keep up-to-date information.

The evaluation criteria of the WS-QoS architecture are:

1. Scalability: The Ws-QoS architecture supports scalability by providing the following components: requirement Manager that retrieves and updates the user's QoS requirements, Web service Broker that selects services dynamically and efficiently, and WS-QoS Monitor that checks the compliance of service offers. The architecture serves a high number of users with assured QoS.
2. Extensibility: The WS-QoS architecture is extensible by providing a standardised XML-based QoS specification, which contains three XML

documents: *WS-QoSRequirementDefinition* that specifies user's QoS requirements, *WSQoSOfferDefinition* that contains the specification of QoS offers and *QoSInf* that holds information on different aspects of QoS properties.

3. Commodity to standards: The architecture is commodity to Web services core standards such as UDDI and WSDL.
4. Ease of implementation: The WS-QoS architecture is implemented using C# and ASP.NET application. The implementation does not consider how the architecture selects the service.
5. Technique for selection: No information about the selection technique.

7.2.6 Web Service QoS Architecture

A Web services QoS architecture (WQA) in [98] extends the current Web service architecture with QoS broker. The user sends a QoS query to the broker then it connects to UDDI registry and collects all the Web services with the similar function. The QoS broker filters the QoS-aware services using an algorithm to choose the optimum services.

The evaluation criteria of the WQA Certifier are:

1. Scalability: The architecture does not support scalability.
2. Extensibility: The architecture extends the current UDDI Inquiry functions with two methods: *find_business_qos* and *find_service_qos*.
3. Commodity to standards: The architecture is commodity to Web services core standards such as UDDI and WSDL.
4. Ease of implementation: No information about the system implementation.
5. Technique for selection: No information about the selection technique.

7.2.7 Comparison between the Quality-Based Web Service Architecture and the Related Architecture

The evaluation criteria of the proposed quality-based Web service architecture (QWSA) are:

1. Scalability: The QWSA architecture supports scalability from the service providers' side that enables them to publish huge number of their services specified with functional specification to UDDI and with quality specifications to quality server. QWSA manages the dynamic nature of the quality criteria that is to keep up-to-date information on quality specifications currently available for services. However, QWSA does not support scalability from the requester's side. Only one requester at a time can request the system. It needs to extend the functionality of the quality server to manage several queries that are sent concurrently by multi- requesters
2. Extensibility: The functionality of the quality server within the QWSA architecture can be extended with a notification mechanism that sends a notification to quality manager of any changes in the quality criteria to keep update information in the quality database. Also, the functionality of the quality server can be extended to manage several queries that are many requesters send their queries concurrently. The quality model, which is based on the architecture, is extensible, that can add more quality sub-criteria within each quality criteria group without altering the selection process. The WSDL is extended with the quality criteria classification to support quality aspects.
3. Commodity to standards: The quality criteria classification is accommodated within existing Web services core specification standards that is WSDL and UDDI. This enhancement is used in the quality matchmaking process (QMP) to select the best Web service.
4. Ease of implementation: The quality matchmaking process, which is based on the QWSA architecture is implemented easily using Windows application

written in C# language in the Visual Studio .NET 2003 environment. The QWSA architecture can be further implemented using Web Service Application in the Visual Studio .NET 2003 environment.

5. Technique for selection: The service selection technique depends on the matchmaking mechanism that is based on the mathematical model. A quality matchmaking process (QMP) is developed in order to select the best service.

Table 7-1 shows comparison result between the proposed quality-based Web service architecture and the related above six architectures. It is seen that the QWSA is best to select because it considers all the evaluation criteria except the scalability one. The only disadvantage of QWSA architecture is that it does not support concurrent huge number of requests. But the architecture is extensible that can support the scalability without having to make major changes to the system infrastructure. So, this disadvantage is required further investigation in the future work.

Table 7-1 Comparison between QWSA and Related Architectures

Architectures	**Evaluation Criteria**				
	Scalability	Extensibility	Commodity to standards	Ease of implementation	Technique for selection
QCWS	Yes	No	No	Yes	Negotiation
UX	Yes	Yes	Yes	No	Keyword matching
Quality Broker Architecture	No	Yes	No	No	Negotiation
QoS Certifier	No	Yes	Yes	No	No
WS-QoS	Yes	Yes	Yes	Yes	No
WQA	No	Yes	Yes	No	No
QWSA	No	Yes	Yes	Yes	matchmaking based on mathematical

					model

7.3 Evaluating the Quality Matchmaking Process

Most of the proposed quality-based Web service selection approaches depend on matchmaking mechanisms. The matchmaking mechanism matches quality requirements of the service requester with the published quality specifications of the service provider. The matchmaking mechanism varies in the previous work from one approach to another, in its simplest form a simple query matching process is used, others using semantics approaches [76], [70], [103], [33], [108], [38] or computation approaches [90], [72].

7.3.1 Semantic Matchmaking Algorithm

The matchmaking algorithm in [76], [70], [103], [33] supports semantic matchmaking between service advertisements and service requirements. Semantic matchmaking is based on DAML-S service description ontology. DAML-S aims to make Web services computer-interoperable and to facilitate Web service discovery. It defines the notions of a Service Profile (what the service does), a Service Model (how the service work) and a Service Grounding (how to use the service). However, this thesis proposes a quality matchmaking process (QMP), which is based on the mathematical model. Also, this thesis uses WSDL description language instead of DAML-S and extends the WSDL with the quality classification.

Maximilien and Singh [38], [108] propose a matchmaking algorithm, which is used to match consumers policies or constraints to advertised service policies. The matchmaking algorithm is divided into four steps: interface matchmaking, policy matchmaking, semantic matchmaking and quality matchmaking. The first step is

to find the services by considering only the interface matchmaking. Next policy matchmaking is performed on the returned list by matching the advertised policy for each service with the required policy. Next the returned list is reduced by applying the semantic matchmaking by semantically match two qualities by considering their relationship to find if they are related. A quality match occurs when the quality type and unit are the same and the required value of the quality is within the range of the advertised quality value.

The proposed quality matchmaking process (QMP) in this thesis consists of four steps: interface matchmaking, quality criteria type matchmaking, quality criteria value matchmaking and mathematical matchmaking. The first step interface matchmaking is similar to the interface matchmaking in [38] and [108] but the remaining steps are different. In the proposed QMP, the quality criteria type matchmaking matches the required quality type such as Availability with the advertised quality type. The quality criteria value matchmaking retrieves the result if the required value is less than or within the range of the advertised quality values. The mathematical matchmaking is the core step in QMP, which is based on the mathematical model to find the best advertised service with a minimum distance.

7.3.2 QoS Computation Algorithm

The quality matchmaking process (QMP) in this thesis is based on the computation approach. There are three approaches which are similar to the proposed QMP, as described below.

The QoS matchmaking algorithm, which is proposed in [72], is based on the QoS computation model. The QoS computation model uses the Euclidean distance measure in order to find the nearest Web service to the QoS specifications of the consumer that is to find a Web service with a minimum Euclidean distance. They normalize the QoS matrix by using maximizing and minimizing equations that considering the type of the QoS parameter. For example, Response Time needs to

be normalized by minimization using the minimizing equation while Availability needs to be normalized by maximization using maximizing equation.

A QoS computational model is presented in [90] for Web service selection. The QoS computation computes the QoS value for each Web service, the higher the value the best the service to select. The QoS criteria for each Web service represents in a matrix Q. The Q matrix is normalized by considering the type of the criteria. The increase of certain criteria benefits the service requester such as availability while the decrease of certain criteria benefits the service requester such as cost criteria.

A service selection approach that are based on QoS computation is presented in [88]. The candidate Web services with different quality criteria values are represented in a matrix Q. Some of the criteria could be negative that is the higher the value the lower the quality such as Execution Time and Price. Other criteria are positive that is the higher the value the higher the quality such as Availability.

The above computation three approaches that used the matchmaking mechanism do not consider the service requester quality preferences of the quality criteria and therefore do not consider the weight or priority of each quality criteria.

The proposed mathematical model uses two methods in order to select the best Web service. Analytical Hierarchy Process (AHP) method is used to calculate the quality criteria weights based on service requester quality preferences. Euclidean distance method is used as in [72], to measure the distance between the quality requirements specified by the service requester and the quality specifications specified by the service provider. The Web service with minimum Euclidean distance is the best service to select.

However, the proposed mathematical model has a drawback that it is only consider the positive quality sub-criteria and not consider the negative criteria as in the above three computation approaches. The positive sub-criteria are Availability and Reputation, which is the higher the value the higher the quality.

The negative quality sub-criterion is the Price, which is the higher the value, the lower the quality. That drawback is noticed through the scenarios mentioned in Section 7.4.1.

7.4 Evaluating the Quality Service Selection System

This section evaluates the quality service selection system (QSSS), which based on the mathematical model and quality classification, through two steps. Firstly, use an Amazon E-Commerce Service (ECS) case study. Secondly, use e-commerce scenarios applied on the ECS case study.

Amazon E-Commerce Service (ECS) is selected as the best Web service as shown scenario 1 in Section 1.1.1. The following sections use ECS Web service to select the best books. The selection is based on the quality matchmaking process (QMP).

7.4.1 Amazon E-Commerce Service Case Study

Amazon E-Commerce Service (ECS) [48] (see Appendix D for details) is an Amazon API (Application Program Interface), which is a set of building blocks made up of routines, protocols, and tools that influence how users interface with the service. ECS publishes a Web Services Description Language (WSDL) document that defines all the available ECS APIs, their parameters and the data that they return. ECS offers applications that retrieve information about a set of products, vendors, and transactions. Requesters can access the ECS using either XML over HTTP (REST) or a remote procedure call API with a Simple Object Access Protocol (SOAP) interface. Both of these methods return structured data (product name, manufacturer, price, etc.) in an XML format.

ECS is used as a case study to retrieve information about the products that are offered by different sellers/vendors with different quality criteria such as product price, seller reputation and product availability. The information is retrieved by sending a REST request to Amazon database. The REST request is sent rather

than the SOAP in this thesis because when sending a simple SOAP request to access ECS, an error appears when running the application at the ECS side when processing the request. A simple ASP.NET Web application is taken from [146] is used SOAP request to access Amazon E-Commerce Service (ECS) (see Appendix E for details).

7.4.1.1 Test Amazon E-Commerce Case Study without the Proposed QSSS

A REST (Representational State Transfer) request as shown in Figure 7-1 is sent to Amazon database through Amazon E-Commerce service (ECS). The requester enter REST request URL (Uniform Resource Locator) into the browser and hit the "Go" button to make the request. The browser will make an HTTP GET request to the server and display the result as shown in Figure 7-2. If the requester is using Internet explorer, the XML data returned by ECS is displayed in readable form.

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService &
SubscriptionId=1NC71HN9R7AE4KJ1G3G2 &Operation=ItemSearch &Title=web
services & SearchIndex=Books &MerchantId=All &ResponseGroup=Item Attributes.
OfferFull
```

Figure 7-1 REST Request to Amazon database

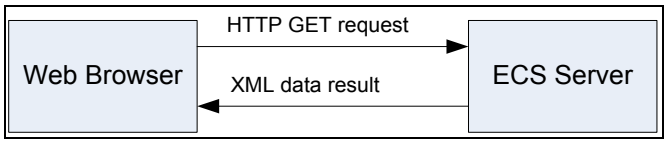

Figure 7-2 Transaction between Requester and Amazon E-Commerce Service

Every REST request to ECS as shown in Figure 7-1: begins with an URL: http://webservices.amazon.com/onca/xml?Service=AWSECommerceService

The URL is followed by a series of parameters separated by an ampersand (&) character. Each parameter consists of a key and a value, separated from each other by an equal sign (=). The parameter and their values are case sensitive.

Figure 7-1 shows an example of a REST request that searches for books about Web services. The parameters in the example are described below:

Table 7-2 Parameters of REST Request

Parameter	Description
SubscriptionId= 1NC71HN9R7AE4KJ1G3G2	Required in all ECS request. The developer/requester must sign up for a subscription ID before he/she can use ECS
Operation= ItemSearch	Required in all ECS request. The *Operation* tells ECS what action it should perform. The operation is *ItemSearch*, which tells ECS to perform a search for products in the Amazon.com catalog that meet particular criteria.
SearchIndex=Books	Required by the *ItemSearch* operation. *SearchIndex* tells the *ItemSearch* operation what type of product to search for. The example searches through the *Books* index. There are many other search indexes available such as Music, Video, Computer, Tools, Software, etc.
Title= Web services	*Title* tells the *ItemSearch* operation to search Amazon.com catalog for specific text value which is in the example *Web services*.
ResponseGroup= ItemAttributes, OfferFull	Specifies what kinds of data are returned in a response. The default response groups in the *ItemSearch* operation are *Request* and *Small*. In the example, the response groups are: *ItemAttributes* and *OfferFull*. These response groups are selected in order to retrieve the following quality criteria: product price, availability and seller reputation. *ItemAttributes* provides information about the book such as its title. *OfferFull* provides information about the product (book) availability, product price and seller ID and nickname.
MerchantId=All	It includes in the request to get availability information for products sold by vendors excluding Amazon. *OfferFull* response provides availability information for products sold by Amazon.

When Service requester sends REST request of Figure 7-1 to ECS database, the interface matchmaking algorithm (step-1 in the quality matchmaking process (QMP)) matches the functional requirements (category “Books” and Title “Web services”) with category of type “Books” in the ECS database and then retrieves all the books contain the keyword “Web services”. The result is retrieved in an XML data format as shown in Figure 7-3. The result contains 933 books and 94 pages as shown in Figure 7-3 from the elements <TotalResults> and

<TotalPages>. Appendix F displays some of the XML data result. The result is further filtered by using the quality criteria type matchmaking algorithm (step-2 in the quality matchmaking process (QMP)). This algorithm retrieves the books associated with quality criteria type provided by the Response Group (Item Attributes, Offer Full) that the requester has specified it when sent the request to ECS (see Figure 7-1). The quality criteria that are provided by Item Attributes and Offer Full response groups are:

Seller reputation: is retrieved from the element *AverageFeedbackRating* as shown in Figure 7-3. Seller reputation value is between 1 and 5, where 5 is the best.

Product price: is retrieved from the element *Price* as shown in Figure 7-3.

Availability: is retrieved from the *Availability* as shown in Figure 7-3. Availability in Amazon E-Commerce Service (ECS) is non-quantitative value such as “Usually ships in 24 hours”, “Limited availability”. To quantify the availability, a percentage value is given to each availability message as shown in Table 7-3. For example, the availability message “Usually ships in 24 hours” gets value from 95-100%.

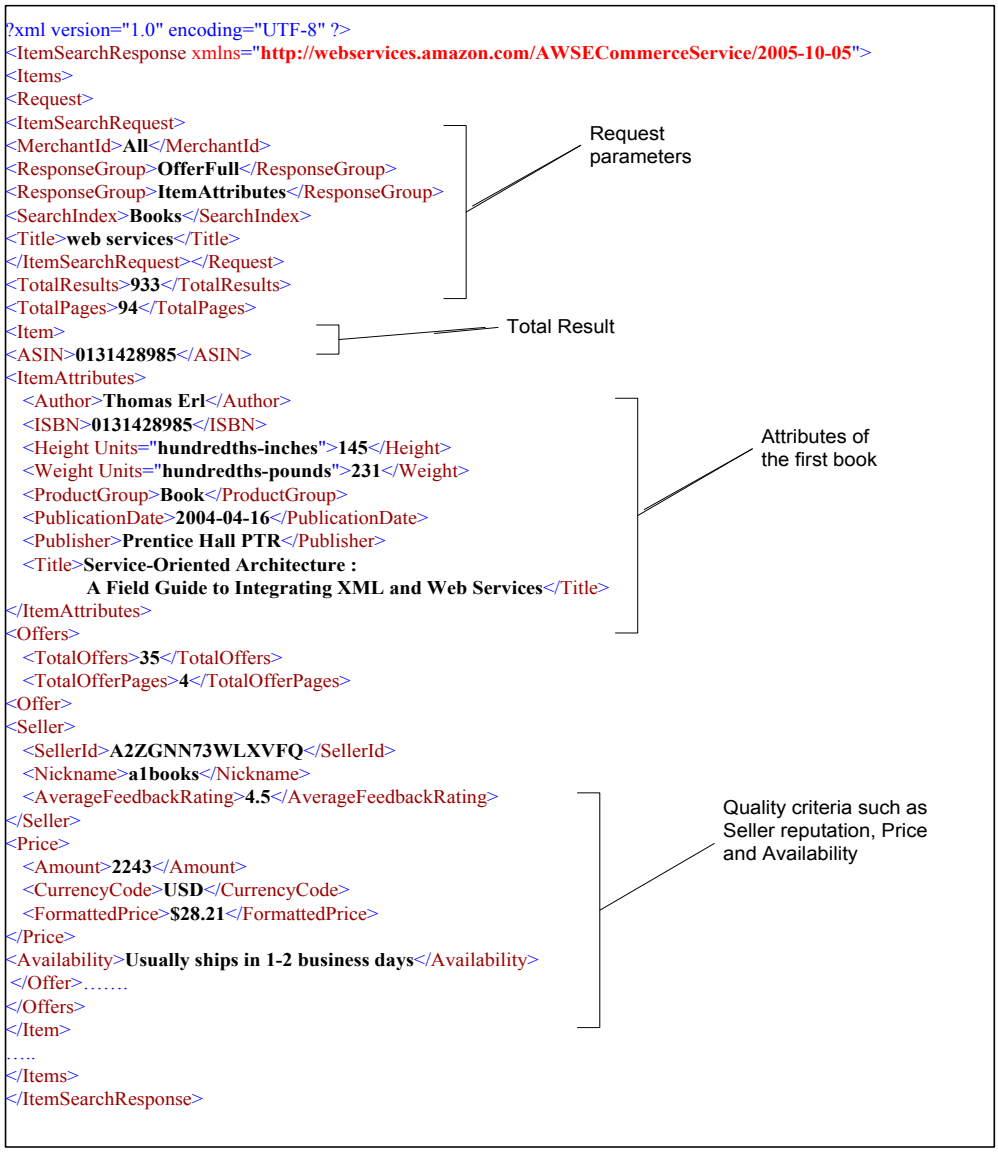

Figure 7-3 XML Data Result of REST Request

Table 7-3 Availability

Availability Value	Availability Element	Description
95-100%	Usually ships in %X	A dynamic response where %X represents a variable amount of time.
85-94%	In stock soon. Order now to get in line. First come, first served.	The item is available for purchase, but is not in stock.
70-84%	Limited Availability	Used for items sold by third-parties if an item is out of stock, but may be available for purchase later.
70-84%	Out of Print—Limited Availability	Customers can choose to be notified if a copy becomes available.
40-69%	Special Order	Titles occasionally go out of print or publishers run out of stock. The buyer is notified if the item becomes unavailable."
0-39%	Not yet released	The item is not available for purchase. The item may or may not have a projected release date.
0-39%	Not yet published	The item is not available for purchase. The item may or may not have a projected release date.
0-39%	This item is not stocked or has been discontinued.	The item is not available for purchase.
0-39%	Out of Stock	The item is currently not available for purchase, but may be in the future.
%X	Only %X left in stock--order soon (more on the way).	The item is available for purchase, but there may only be a few copies left where %X represents a variable amount of time.
%X	Only %X left in stock--order soon.	The item is available for purchase, but there may only be a few copies left where %X represents a variable amount of time.

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService &
SubscriptionId=1NC71HN9R7AE4KJ1G3G2&AssociateTag=webservice1-20
&Operation=SellerLookup &SellerId=A3E0GMZ4YFS6AQ & ResponseGroup=Seller
```

Figure 7-4 REST Request for Retrieving Seller Information

The request in Figure 7-1 doesn't provide information about the sellers. However, another request is needed as shown in Figure 7-4**,** which includes the following parameters:

Table 7-4 Parameters of REST Request

Parameter	Description
SubscriptionId= 1NC71HN9R7AE4KJ1G3G2	It is same as the parameter in request shown in Figure 7-1
AssociateTag=webservice1-20	Amazon's associate includes Web site owner, Amazon seller or Web developer. The associates must sign up for an associate tag before using ECS.
Operation=SellerLookup	The *SellerLookup* operation allows the requester to retrieve information related to specific vendors' feedback from customers, rating, location and name. The rating is returned in the *Seller/AverageFeedbackRating* element and it is equivalent to seller reputation criteria which its value is between 1 and 5.
SellerId=A3E0GMZ4YFS6AQ	*SellerId* values are retrieved from the first request of Figure 7-1. The request in Figure 7-4 looks up for up to five Seller IDs by inserting commas between them.
ResponseGroup=Seller	The *Seller* response group provides the seller ID, nickname, average feedback rating which is equivalent to the seller reputation and location for each seller.

When send REST request of Figure 7-4, the XML data is returned as shown in Figure 7-5. It provides information about a specific seller by retrieving the seller ID from the result of Figure 7-3. The information retrieved is:

Seller URL: is retrieved from the element *GlancePage* as shown in Figure 7-5.

Seller Reputation: is retrieved from the element *AverageFeedbackRating* as shown in Figure 7-5 . Seller reputation value is calculated from the equation: $q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$ where R_i is the customer's feedback rating on the seller, n is the number of times the seller has been graded. The value of seller reputation is between 1 and 5, where 5 is the best.

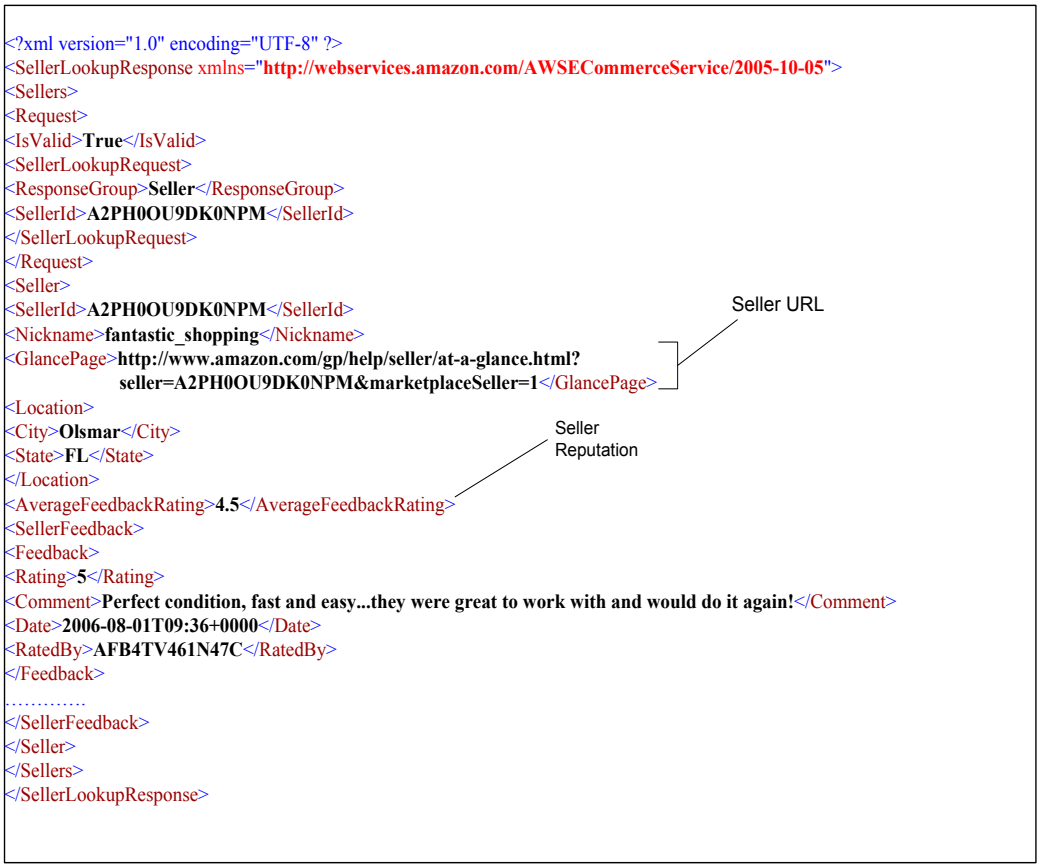

Figure 7-5 XML Data Result of REST Request of the seller

The information retrieved from the XML data results (as shown in Figure 7-3 and Figure 7-5) is organized in Appendix G into Table G-1; some of the data is shown in Table 7-5. Table G-1 shows nine books with 76 different sellers and different quality criteria values. The service requester can't easily select manually the preferred book among 76 options in Table G-1, so he/she needs a technique in order to assist him/her to select the preferred book in automated way. This thesis proposes a quality matchmaking process (QMP) depends on requester quality preferences and requirements in order to select the best book. The QMP is illustrated by developing a quality service selection system (QSSS) and Table G-1 is used as a database in the QSSS as described in the coming section.

Table 7-5 Amazon ECS database

Product Name	Seller Name	Availability	Price	Seller Reputation	Seller URL
Service-Oriented Architecture	hebertbooks	99	24.1	3.4	http://www.amazon.com/seller=A1RAFT0AR298LX
	fantastic_shopping	87	24.14	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	fun-for-all58	75	24.3	3.5	http://www.amazon.com/seller=A1MD3EN9VM2K1F
	yaleiz	80	27.99	4	http://www.amazon.com/seller=A1MOV0BA9DKUFU
	a1books	97	28.21	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	Amazon.com	99	28.34	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	allnewbooks	80	29.19	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	95	30.07	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
Web Services Platform Architecture	fantastic_shopping	90	29.94	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	amz_book	78	29.95	3	http://www.amazon.com/seller=A3B9364CV8QDO9
	a1books	98	31.05	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	Amazon.com	99	31.49	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	allnewbooks	65	32.34	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	30	33.41	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	thebookrackrh	75	34.09	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	a1books_nj	95	34.11	3.6	http://www.amazon.com/seller=A3E0GMZ4YFS6AQ
	alphacraze	82	34.34	3.4	http://www.amazon.com/seller=A2NT0F3A6LH7YD
	alphacrazeoutlet	97	34.34	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1
J2EE Web Services	bookbensara	95	29.75	4.2	http://www.amazon.com/seller=A3H8H6KI3KCVA5
	fantastic_shopping	85	34.63	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	Amazon.com	99	34.64	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	a1books	98	35.04	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	thebookrackrh	80	35.4	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	allnewbooks	95	35.49	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	20	37.2	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	alphacrazeoutlet	79	37.93	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1
	alphacraze	99	37.93	3.4	http://www.amazon.com/seller=A2NT0F3A6LH7YD
	a1books_nj	96	38.51	3.6	http://www.amazon.com/seller=A3E0GMZ4YFS6AQ

7.4.1.2 Test Amazon E-Commerce Case Study with the Proposed QSSS

In the previous section, it is seen that when sending a request to Amazon E-Commerce Service (ECS), the result contains 933 books as shown in Figure 7-3. The result is large enough that the service requester can't easily select the best service or book manually.

To overcome the above limitation, this thesis develops a quality service selection system (QSSS) (see Chapter 6), which enables the service requester to specify his/her quality preferences and requirements and assists the requester to select the best service automatically. QSSS implements the quality matchmaking process

(QMP), which described in Section 5.5 . The QSSS technique is based on the mathematical model, quality classification and the requester quality preferences and requirements.

Some of the returned result of REST requests in Figure 7-1 and Figure 7-4, is saved in Table G-1 (see Appendix G for details). Table G-1 contains the following fields: *ProductName*, *SellerName, Availability*, *Price*, *SellerReputation* and *SellerUR*L, and contains 76 records or books with different quality criteria values. The *ProductName* field is the title of the books, *SellerName* is the name of the book seller, *Availability* is the book's availability and its ready for shipment, *Price* is the price of the book offered by the seller, *SellerReputation* is the reputation of the book seller and is based on the requester feedback and *SellerUR*L is the Web site location of the sellers that the requester after selecting the best book he/she contacts the seller in order to buy it. The *Availability, Price* and *SellerReputation* are considered as Availability sub-criteria within the Failure Probability, Service Price sub-criteria within the Cost group and Reputation sub-criteria within the Trustworthiness group respectively.

The requester wanted to select the best book from the books which is saved in Table G-1 using the QSSS. The scenarios below demonstrate the books selection based on different requester quality preferences and requirements.

7.4.1.3 Applying Quality Service Selection System to Use Case Scenarios

The quality service selection system (QSSS) is based on the quality classification and the mathematical model. The quality classification, which is explained in Chapter 3 consists of quality criteria groups: Performance, Failure probability, Trustworthiness and Cost. Each of these quality criteria contains number of sub-criteria. QSSS enables the service requester to select freely his/her preferred quality criteria group and the corresponding sub-criteria. The service requester specifies the quality requirements from two perspectives: Web service and the

services or products provided from the corresponding selected Web service. These two perspectives are described is Section 3.3.

This section shows scenarios in two levels. The first scenario illustrates the selection of Web service as in scenario 1 and the remaining scenarios illustrate the selection of services or products provided by the corresponding selected Web service.

Scenario 1: Web service selection

The requester looks for a search engine Web services to search for books. There are four Web services as shown in Table 7-6: Amazon E-Commerce Web Services (ECS), Google Web Service, eBay Web Service and Yahoo Web service.

These four Web services have the same functionality that it enables the requester to search for products or items. However, there is no criterion to differentiate between them, so the quality criteria is an important factor to differentiate between them. Also, it is not easy for the requester to select manually the best Web service with different quality criteria values, so it requires a way that enables the requester selects the best Web service automatically. The QSSS system enables the service requester to select the best Web service in an automated way as shown below.

The requester uses QSSS system to select the best Web service with the following requirements:

- Throughput is six times more important than the Availability.
- Throughput is three times more important than the Price.
- Price is two times more important than the Availability.
- The requirement value of Throughput: High, Availability : High and Price :

Low.

The selected above quality criteria (Throughput, Availability and Price) is from the Web service's perspective and based on the quality criteria classification (See Section 3.3). Table 7-7 shows the four Web services (Amazon, Google, eBay and Yahoo) with its corresponding quality criteria.

Table 7-6 Web Service Description

Web Services	Description
Amazon ECS	• Search catalogue, retrieve product information, images and customers reviews. • Search sellers and offers.
eBay Web Service	• View information about items listed on eBay. • Retrieve lists of items a particular user is currently selling through eBay. • Provide feedback about other users at the conclusion of a ecommerce transaction.
Yahoo Web Service	Enables developers, businesses and researchers to search for products and services in a powerful way.
Google Web Service	• Search Web pages. • Get information about search result including URL, title and directory category.

Table 7-7 Web services

Quality Criteria	Web Services			
	Amazon	**Google**	**eBay**	**Yahoo**
Throughput/day	2200	1000	1440	1200
Availability	98	98	95	90
Price/month ($)	0	0	5	0

After applying the mathematical model, which is described in Chapter 5, the weight of the quality criteria is:

$$W = [0.667 \quad 0.111 \quad 0.222]$$

It is noticed that Throughput criteria is the most important criteria which has the highest priority (0.667) then the Price (0.222) and the last is the Availability (0.111).

The output result that is based on the requester's quality requirements and preferences is shows in Table 7-8. It is seen that Amazon Web service (ECS) is the best one to select because its matching distance is the minimum "0.167". So ECS is the best Web service that the requester can select. The output displays the quality criteria values for each Web service in order enable the requester judge if that the Web service with the minimum distance satisfies his/her requirements. If the result does not satisfy his/her expectation, then he/she can specify another quality preferences and requirements.

Table 7-8 Output of Web Service Selection

Web Services	Matching Distance	Throughput	Availability	Price
Amazon	**0.167**	**2200**	**98**	**0**
eBay	**0.628**	**1440**	**95**	**5**
Yahoo	**0.852**	**1200**	**90**	**0**
Google	**1.115**	**1000**	**98**	**0**

Scenario 2:

After selection Amazon E-Commerce Service (ECS) as a best Web service, the requester wants to search ECS to select a book regarding to its availability, seller reputation and its price. When sending a REST request to ECS database, it is

noticed that the output result contains a huge number of books about 933 , which is not easy for the requester to select manually the best book with different quality criteria values, so it requires a way that enables the requester selects the best book automatically. The QSSS system enables the service requester to select the best Web service in an automated way as shown below.

The requester specifies his/her quality requirements using QSSS system as in the following:

1. The service requester selects the quality criteria with the following preferences or importance:

- Failure probability is assigned by the service requester as five times more important than the Trustworthiness.
- Failure probability is assigned by the service requester as three times more important than the Cost.
- Cost is assigned by the service requester as two times more important than the Trustworthiness.

2. The service requester specifies the sub-criteria requirement values as in the following:

- Requirement value of Availability sub-criterion value, which is included in the Failure probability criteria group, is equal “High”.
- Requirement value of reputation sub-criterion value, which is included in the Trustworthiness criteria group, is equal “Medium”.
- Requirement value of Service Price sub-criterion value, which is included in the Cost criteria group, is equal “Medium”.

The requester specifies the quality preferences or importance from the *Preference Selection* form, and specifies the sub-criteria quality values from the *Requirements Value* form as described in Section 6.3.2.

From the input values: the quality criteria preferences and the quality sub-criteria requirement levels specified by the requester above, QSSS calculates the following:

1. The pair-wise comparison matrix A is formed by creating an instance of a Matrix class and calling *FillMatri*x() method from *Utilities* class. The matrix A is formed based on requester preferences values as in the following:

$$A = \begin{array}{c} \\ FP \\ T \\ C \end{array} \begin{array}{c} \begin{array}{ccc} FP & T & C \end{array} \\ \begin{bmatrix} 1 & 5 & 3 \\ 0.2 & 1 & 0.5 \\ 0.333 & 2 & 1 \end{bmatrix} \end{array}$$

2. The weights of quality criteria can be calculated from the matrix A calling *CalculateWeights*( ) method from *Utilities* class. The weights vector of quality criteria is:

$$W = \begin{bmatrix} 0.648 & 0.122 & 0.23 \end{bmatrix}$$

The total weight is equal to 1:

$$W(FP) + W(T) + W(C) = 1$$

It is noticed that Failure probability criteria is the most important criteria which has the highest priority (0.648) then the Cost (0.23) and the last is the Trustworthiness (0.122).

Because the requester selects only one sub-criterion in each quality criteria group, the weight of each sub-criterion is equal to the weight of its criteria group that is the weight of the Availability (AV) sub-criteria is equal to he weight of the Failure probability (FP) weight (0.648), the weight of the Reputation (REP) sub-criteria is equal to the weight of Trustworthiness (T) (0.122) and The weight of the Service Price(SP) sub-criteria is equal to the weight of Cost (C) (0.23).

3. The Consistency Ratio (CR) measures the degree of consistency of the selected preferences values of the quality criteria. The Consistency Ratio (CR) is calculated by calling *ConsistencyRatio*( ) method from Utilities class, The Consistency Ratio (CR) is equal to 0.0032 which is less than 0.1, so the pair-wise requester preferences is consistent and the procedure will continue to select the best book.

4. The performance matrix P is retrieved by sending an SQL query as shown in Listing 7-1, to an MS-Access database which contains information about the books (see Table G-1). The query matches the sub-criteria requirement levels (High Availability, Medium Seller Reputation and Medium Price) with the books in the MS-Access database as shown in Table G-1. The result of the SQL query is shown in Table 7-9.

```
if(Availability.Equals("High")&&Reputation.Equals("Medium")&&ServicePrice.Equals("Medium"))
  {
   oleDbDataAdapter1.SelectCommand.CommandText="SELECT ProductName,SellerName,SellerURL,
   Availability,Price,SellerReputation FROM AmazonTable WHERE AmazonTable.Availability
   BETWEEN 80 AND 100 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 3.9 AND
   AmazonTable.Price BETWEEN 25 AND 49.99";
  }
```

Listing 7-1 SQL Query

Table 7-9 SQL Query Result Obtained for Performance Matrix

Product Name	Seller Name	Availability	Price	Seller Reputation
Service-Oriented Architecture	allnewbooks	80	29.19	2.6
Web Services Platform Architecture	a1books_nj	95	34.11	3.6
Web Services Platform Architecture	alphacraze	82	34.34	3.4
Web Services Platform Architecture	alphacrazeoutlet	97	34.34	3.7
J2EE Web Services	thebookrackrh	80	35.4	2.8
J2EE Web Services	allnewbooks	95	35.49	2.6
J2EE Web Services	alphacraze	99	37.93	3.4
J2EE Web Services	a1books_nj	96	38.51	3.6
How to Break Web Software	powells_books	90	34.99	2.8
Core Security Patterns	fun-for-all58	98	33.85	3.5
Core Security Patterns	allnewbooks	85	38.64	2.6
Core Security Patterns	thebookrackrh	97	39.2	2.8
Core Security Patterns	amz_book	84	39.95	3
Building Web Services with Java	allnewbooks	82	32.34	2.6
Building Web Services with Java	alphacraze	97	34.34	3.4
Understanding SOA with Web Services	superbookdeals	96	25.04	2.5
Understanding SOA with Web Services	amz_book	95	25.95	3
Understanding SOA with Web Services	a1books_nj	86	27.28	3.6
Understanding SOA with Web Services	lphacrazeoutlet	98	27.39	3.7

The performance matrix is created from the SQL query result (see Table 7-9) as the following:

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19
AV	80	95	82	97	80	95	99	96	90	98	85	97	84	82	97	96	95	86	98
P=REP	2.6	3.6	3.4	3.7	2.8	2.6	3.4	3.6	2.8	3.5	2.6	2.8	3	2.6	3.4	2.5	3	3.6	3.7
P	29.19	34.11	34.34	34.34	35.4	35.49	37.93	38.51	34.99	33.85	38.64	39.2	39.95	32.34	34.34	25.04	25.95	27.28	27.39

5. The requirement values r for the sub-criteria, which is selected by the requester is shown below:

$$r = \begin{matrix} AV \\ REP \\ P \end{matrix} \begin{bmatrix} High \\ Medium \\ Medium \end{bmatrix} = \begin{bmatrix} 98 \\ 3 \\ 40 \end{bmatrix}$$

The requirement level for the Availability is "High" which its value equals 98 and located within its range [80-100]. The requirement level for Reputation is "Medium" which is equal 3 and located within its range [2.5-3.9]. The requirement level for Price is "Medium" which is equal 40 and located within its range [25-49.99]. The aforementioned values for the requirement value: High, Medium and low depends on the service domain and on the type of the sub-criteria. For example, the requirement value for buying a book is different than buying a computer and the requirement value of Availability is different than Reputation.

6. Euclidean distance is calculated by calling E*uclideanDistance* ( ) method from *Utilities* class.

Table 7-10 Output Result of Scenario2

Product Name	Seller Name	Matching distance	Availability	Reputation	Price
J2EE Web Services	alphacraze	0.44	99	3.4	37.93
Core Security Patterns	thebookrackrh	0.452	97	2.8	39.2
Core Security Patterns	fun-for-all58	0.454	98	3.5	33.85
J2EE Web Services	a1books_nj	0.459	96	3.6	38.51
Web Services Platform Architecture	alphacrazeoutlet	0.46	97	34.34	3.7
Building Web Services with Java	alphacraze	0.461	97	34.34	3.4
Understanding SOA with Web Servic	lphacrazeoutlet	0.466	98	27.39	3.7
J2EE Web Services	allnewbooks	0.475	95	35.49	2.6
Web Services Platform Architecture	a1books_nj	0.476	95	34.11	3.6
Understanding SOA with Web Servic	superbookdeals	0.487	96	25.04	2.5
Understanding SOA with Web Servic	amz_book	0.492	95	25.95	3
How to Break Web Software	powells_books	0.514	90	34.99	2.8
Core Security Patterns	allnewbooks	0.545	85	38.64	2.6
Core Security Patterns	amz_book	0.549	84	39.95	3
Understanding SOA with Web Servic	a1books_nj	0.565	86	27.28	3.6
Web Services Platform Architecture	alphacraze	0.583	82	34.34	3.4
Building Web Services with Java	allnewbooks	0.59	82	32.34	2.6
J2EE Web Services	thebookrackrh	0.599	80	35.4	2.8
Service-Oriented Architecture	allnewbooks	0.619	80	29.19	2.6

From the output result in Table 7-10, the first book with title "J2EE Web Services" and its provider is "alphacraze" is the best book to select because its matching distance is the smallest (0.44). It is reasonable that the first book is the best because it has the highest Availability value (99) and the required Availability has the highest priority (0.648). The output displays the quality criteria values for each Web service in order enable the requester judge if that the Web service with the minimum distance satisfies his/her expectations. If the result does not satisfy his/her expectation, then he/she can specify another quality preferences and requirements.

Scenario 3:

A requester wants to select a book regarding to its availability, seller reputation and its price constraint. The book's reputation is the most important from the requester's point-of-view, which has the highest priority then the availability and the last important is the price. Also, the requester wants a book with high availability, medium seller reputation and medium book's price.

1. The service requester selects the quality criteria with the following preferences or importance:

- Trustworthiness is assigned by the service requester as two times more important than the Failure probability.
- Failure probability is assigned by the service requester as four times more important than the Cost.
- Trustworthiness is assigned by the service requester as seven times more important than the Cost.

2. The service requester specifies the sub-criteria requirement l as in scenario 1 as the following:

- Requirement value of Availability sub-criterion value, which is included in the Failure probability criteria group, is equal “High”.
- Requirement value of reputation sub-criterion value, which is included in the Trustworthiness criteria group, is equal “Medium”.
- Requirement value of Service Price sub-criterion value, which is included in the Cost criteria group, is equal “Medium”.

The requester specifies the quality preferences or importance from the *Preference Selection* form, and specifies the sub-criteria quality values from the *Requirements Value* form as described in Section 6.3.2.

From the input values which specified by the requester above, QSSS calculates the following:

1. The pair-wise comparison matrix A is formed based on requester’s preferences values as in the following:

$$A = \begin{array}{c} \\ FP \\ T \\ C \end{array} \begin{array}{c} \begin{array}{ccc} FP & T & C \end{array} \\ \begin{bmatrix} 1 & 0.5 & 4 \\ 2 & 1 & 7 \\ 0.25 & 0.143 & 1 \end{bmatrix} \end{array}$$

2. The weights vector of quality criteria are calculated from the matrix A as in the following:

$$W = [0.315 \quad 0.602 \quad 0.082]$$

The Reputation sub-criterion is the most important criterion which has the highest priority (0.602) then the Availability (0.315) and the last is the Service Price (0.082).

3. The Consistency Ratio (CR) is equal to 0.002 which is less than 0.1, so the pair-wise requester's judgement is consistent and therefore the procedures will continue in order to select the best book.

4. The performance matrix P is retrieved by sending an SQL query as in step 4 in scenario1, to an MS-Access database which contains information about books (see Table G-1). The result of the SQL query is shown in Table 7-10. The performance matrix P is

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19
AV	80	95	82	97	80	95	99	96	90	98	85	97	84	82	97	96	95	86	98
P=REP	2.6	3.6	3.4	3.7	2.8	2.6	3.4	3.6	2.8	3.5	2.6	2.8	3	2.6	3.4	2.5	3	3.6	3.7
P	29.19	34.11	34.34	34.34	35.4	35.49	37.93	38.51	34.99	33.85	38.64	39.2	39.95	32.34	34.34	25.04	25.95	27.28	27.39

5. The requirement values r for the sub-criteria, which is selected by the requester is shown below:

$$r = \begin{matrix} AV \\ REP \\ P \end{matrix} \begin{bmatrix} High \\ Medium \\ Medium \end{bmatrix} = \begin{bmatrix} 98 \\ 3 \\ 40 \end{bmatrix}$$

6. The Euclidean distance is calculated as in step-5 in scenario1.

7. shows the Output result, which is based on requester's preferences and it is ranked from the smallest matching distance to the largest one.

From the output result as shown in Table 7-13, the first book with title "Understanding SOA with Web Services" and its provider is "alphacrazeoutlet" is the best book to select because its matching distance is the smallest (0.245). It is reasonable that the first book is the best because it has the highest Reputation value (3.7) as seen in Table 7-13 and the required Reputation has the highest priority (0.602).

Table 7-11 Output Result of Scenario3

Product name	Product provider	Matching distance	Availability	Reputation	Price
Understanding SOA with Web Services	alphacrazeoutlet	0.245	98	3.7	27.39
Understanding SOA with Web Services	superbookdeals	0.25	96	2.5	25.04
Core Security Patterns	fun-for-all58	0.251	98	3.5	33.85
J2EE Web Services	alphacraze	0.254	99	3.4	37.93
Understanding SOA with Web Services	amz_book	0.254	95	3	25.95
Web Services Platform Architecture	alphacrazeoutlet	0.255	97	3.7	34.34
Building Web Services with Java	alphacraze	0.255	97	3.4	34.34
Web Services Platform Architecture	a1books_nj	0.261	95	3.6	34.11
Core Security Patterns	thebookrackrh	0.261	97	2.8	39.2
J2EE Web Services	allnewbooks	0.262	95	2.6	35.49
J2EE Web Services	a1books_nj	0.263	96	3.6	38.51
How to Break Web Software	powells_books	0.278	90	2.8	34.99
Understanding SOA with Web Services	a1books_nj	0.289	86	3.6	27.28
Core Security Patterns	allnewbooks	0.298	85	2.6	38.64
Core Security Patterns	amz_book	0.302	84	3	39.95
Web Services Platform Architecture	alphacraze	0.307	82	3.4	34.34
Building Web Services with Java	allnewbooks	0.307	82	2.6	32.34
Service-Oriented Architecture	allnewbooks	0.315	80	2.6	29.19
J2EE Web Services	thebookrackrh	0.315	80	2.8	35.4

The output displays the quality criteria values for each Web service in order enable the requester judge if that the Web service with the minimum distance satisfies his/her requirements. If the result does not satisfy his/her expectation, then he/she can specify another quality preferences and requirements.

Scenario 4:

A requester wants to select a book regarding to its availability, seller reputation and its price constraint. The book's price is the most important from the requester's point-of-view, which has the highest priority then the availability and the last important is the seller reputation. Also, the requester wants a book with high availability, medium seller reputation and medium book's price.

1. The service requester selects the quality criteria with the following preferences or importance:

- Failure probability is assigned by the service requester as four times more important than the Trustworthiness.
- Cost is assigned by the service requester as three times more important than the Failure probability.
- Cost is assigned by the service requester as nine times more important than the Trustworthiness.

2. The service requester specifies the sub-criteria requirement values as in scenario 1 as the following:

- Requirement value of Availability sub-criterion value, which is included in the Failure probability criteria group, is equal “High”.
- Requirement value of reputation sub-criterion value, which is included in the Trustworthiness criteria group, is equal “Medium”.
- Requirement value of Service Price sub-criterion value, which is included in the Cost criteria group, is equal “Medium”.

The requester specifies the quality preferences or importance from the *Preference Selection* form, and specifies the sub-criteria quality values from the *Requirements Value* form as described in Section 6.3.2.

From the input values which specified by the requester above, QSSS calculates the following:

1. The pair-wise comparison matrix A is formed based on requester’s preferences values as in the following:

$$A = \begin{array}{c} \\ FP \\ T \\ C \end{array} \begin{array}{c} \begin{array}{ccc} FP & T & C \end{array} \\ \begin{bmatrix} 1 & 4 & 0.333 \\ 0.25 & 1 & 0.111 \\ 3 & 9 & 1 \end{bmatrix} \end{array}$$

2. The weights vector of quality criteria are calculated from the matrix A as in the following:

$$W = \begin{bmatrix} 0.251 & 0.069 & 0.68 \end{bmatrix}$$

The book Price sub-criterion is the most important criterion which has the highest priority (0.68) then the Availability (0.251) and the last is the seller Reputation (0.069).

3. The Consistency Ratio (CR) is equal to 0.007 which is less than 0.1, so the pair-wise requester's judgement is consistent and therefore the procedures will continue in order to select the best book.

4. The performance matrix P is retrieved by sending an SQL query as in step 4 in scenario1, to an MS-Access database which contains information about books (see Table G-1). The result of the SQL query is shown in Table 7-6. The performance matrix P is

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19
AV	80	95	82	97	80	95	99	96	90	98	85	97	84	82	97	96	95	86	98
P=REP	2.6	3.6	3.4	3.7	2.8	2.6	3.4	3.6	2.8	3.5	2.6	2.8	3	2.6	3.4	2.5	3	3.6	3.7
P	29.19	34.11	34.34	34.34	35.4	35.49	37.93	38.51	34.99	33.85	38.64	39.2	39.95	32.34	34.34	25.04	25.95	27.28	27.39

5. The requirement values r for the sub-criteria, which is selected by the requester is shown below:

$$r = \begin{matrix} AV \\ REP \\ P \end{matrix} \begin{bmatrix} High \\ Medium \\ Medium \end{bmatrix} = \begin{bmatrix} 98 \\ 3 \\ 40 \end{bmatrix}$$

6. The Euclidean distance is calculated as in step-5 in scenario1.

7. Table 7-12 shows the Output result, which is based on requester's preferences and it is ranked from the smallest matching distance to the largest one.

From the output result as shown in Table 7-12, the first book with title "J2EE Web Services" and its provider is "alphacraze" is the best book to select because its matching distance is the smallest (0.192). It is noticed that the first book is the

best because it has the minimum Euclidean distance (0.192). However, the Service Price as seen in Table 7-15 (37.93) of the best service is not the minimum price.

Table 7-12Output Result of Scenario 4

Product Name	Seller Name	Matching distance	Availability	Reputation	Price
J2EE Web Services	alphacraze	0.192	99	3.4	37.93
Web Services Platform Architecture	alphacrazeoutlet	0.196	97	3.7	34.34
J2EE Web Services	a1books_nj	0.197	96	3.6	38.51
Core Security Patterns	fun-for-all58	0.197	98	3.5	33.85
Understanding SOA with Web Services	alphacrazeoutlet	0.199	98	3.7	27.39
Building Web Services with Java	alphacraze	0.202	97	3.4	34.34
Web Services Platform Architecture	a1books_nj	0.205	95	3.6	34.11
Core Security Patterns	thebookrackrh	0.213	97	2.8	39.2
Understanding SOA with Web Services	amz_book	0.229	95	3	25.95
J2EE Web Services	allnewbooks	0.23	95	2.6	35.49
Understanding SOA with Web Services	superbookdeals	0.24	96	2.5	25.04
How to Break Web Software	powells_books	0.244	90	2.8	34.99
Understanding SOA with Web Services	a1books_nj	0.251	86	3.6	27.28
Core Security Patterns	amz_book	0.256	84	3	39.95
Web Services Platform Architecture	alphacraze	0.264	82	3.4	34.34
Core Security Patterns	allnewbooks	0.266	85	2.6	38.64
J2EE Web Services	thebookrackrh	0.288	80	2.8	35.4
Building Web Services with Java	allnewbooks	0.29	82	2.6	32.34
Service-Oriented Architecture	allnewbooks	0.305	80	2.6	29.19

The output displays the quality criteria values for each Web service in order enable the requester judge if that the Web service with the minimum distance satisfies his/her requirements. If the result does not satisfy his/her expectation, then he/she can specify another quality preferences and requirements.

7.4.1.4 Web Service Composition Scenario

Web service composition is the creation of a Web process from individual Web services.

The proposed quality service selection system (QSSS) can be published as a Web service and used in the Web service composition [147] as shown Figure 7-6.

The following scenario explains how the QSSS which is called *SelectProduct* service in Figure 7-6 is involved in the Web service composition.

Scenario

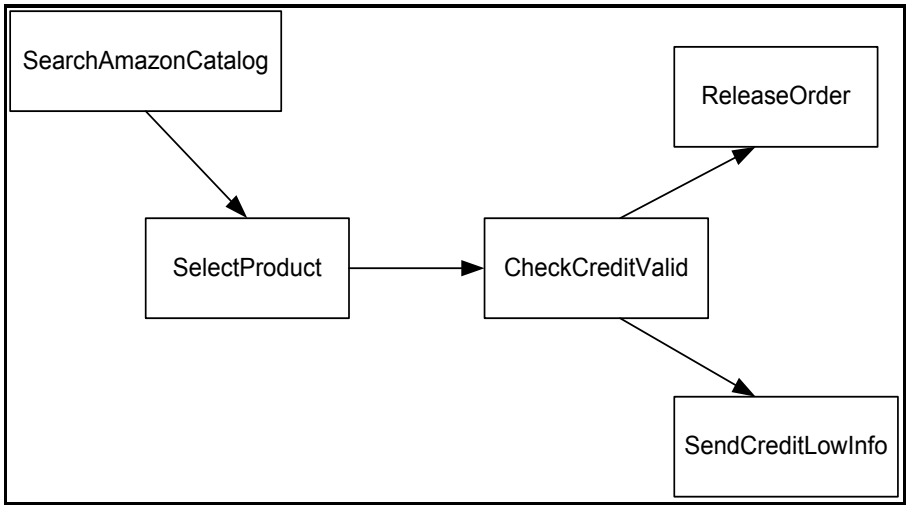

Figure 7-6 Web Service Composition using QSSS

Figure 7-6 shows a Web service composition using QSSS in buying a book in Amazon.com. The Web services involved in this process are: *SearchAmazonCatalog*, *SelectProduct*, *CheckCreditValid*, *ReleaseOrder* and *SendCreditLowInfo.*The *SearchAmazonCatalog* service is used to search Amazon database within the Book catalogue for a certain book title. The *SelectProduct* service selects the best product (book) based on quality criteria classification and the mathematical model as described in Chapter 6. After the requester selects the desired book, the requester's credit card is checked for validation using the *CheckCreditValid* service. If the *CheckCreditValid* service returns success, the *ReleaseOrder* service is invoked to send the book(s), else the *SendCreditLowInfo* service is invoked to give information that the credit card is invalid.

7.5 Discussion

The efficiency of the quality service selection system (QSSS) has been introduced by comparing the book selection from the Amazon E-Commerce (ECS) database without applying the QSSS and the book selection from ECS with applying the QSSS. It is noticed in Section 7.4 that when sending a REST request to ECS, the output result contains 933 books and 94 pages as shown in Figure 7-3. There are 76 books out of 933 are selected and organized in Table G-1 in order to illustrate the quality matchmaking process (QMP) in a simple way. The 76 books are offered by different sellers and with different quality criteria values. The service requester can't easily select manually the preferred book among 76 books in Table G-1 regarding his/her quality preferences, so he/she needs a technique to assist him/her to select the preferred book in an automated way. To solve the aforementioned problem, this thesis has developed QSSS to enable the requester to specify his/her quality preferences and requirements. It calculates the distance between the requester's quality requirements and the books quality specifications saved in the database. The book with the minimum distance is the best book to select.

It is noticed that when implementing different scenarios with different requester's quality preferences and different sub-criteria requirement levels as described in Section 7.4, the mathematical model when normalizing the performance matrix P considers the monotonically increasing sub-criteria and does not consider the monotonically decreasing. The monotonically increasing sub-criteria are that the increasing of criteria benefits the service requester such as Availability and Reputation. The monotonically decreasing sub-criteria are the decreasing of criteria benefits the service requester such as the Service Price.

When the service requester selects the Availability and Reputation with the highest weight then the best service with the minimum Euclidean distance has the highest value of Availability and Reputation, which is desirable by the requester, as shown in scenarios 1, 2 and 3. However, when the service requester selects

Service Price with the highest weight then the best service with the minimum Euclidean distance has the maximum price, which is not desirable by the requester as shown in scenario 4.

There are two factors that affect the service selection approach:

- The relative weights assigned to the quality criteria. Each service requester has preferences between 1 and 9, that biases it toward certain quality criteria and therefore make these criteria weight more than others for a specific domain.
- The requirement values assigned to the quality sub-criteria. The service requester specifies his/her requirement value (High, Medium or Low) for each sub-criterion which affects with the combination of the sub-criterion weight the Euclidean distance and therefore selecting the best matching service that has the minimum Euclidean distance.

In summery, the QSSS has the following advantages:

- It is a combination of subjective (based on requester's preferences and selecting the quality sub-criteria requirement values) and objective (using mathematical method) methods to select the best candidate Web service.
- It is a generic approach because it is based on generic quality classification and it is applied on any service domain.
- It is extensible that new quality criteria group and sub criteria can be added without affecting the mathematical model and the selection technique.
- It is friendly and easy to use that can requesters specify his/her quality preferences and requirements easily.

QSSS has the following disadvantages:

- It is using Analytical Hierarchy Process (AHP) which is subjective method that depends on requester's quality preferences, so it is subject to human error.

- The mathematical model does not consider the decreasing quality sub-criteria such as the Service Price, in calculating the Euclidean distance.

7.6 Summary

This chapter has evaluated the proposed quality-based Web Service Architecture (QWSA), the quality matchmaking process (QMP) approach and the quality service selection system (QSSS).

The QWSA is evaluated by comparing it with the related architectures regarding five criteria: scalability, extensibility, commodity to standards, ease of implementation and technique for selection. It is noticed that the QWSA is the best among the related architectures because it considers all the evaluation criteria except the scalability one. The only disadvantage of QWSA architecture is that it does not support concurrent huge number of requests. But the architecture is extensible that can support the scalability without having to make major changes to the system infrastructure.

The QMP is evaluated by comparing it with the related approaches. It is seen that most of the related service selection approaches depend on matchmaking mechanisms. The matchmaking mechanism varies in the previous works from semantics approaches to computation approaches. The service selection approach in this thesis depends on the quality matchmaking process (QMP). The QMP is based on the mathematical model and it considers the service requester's quality preferences. The related computation approaches do not consider the service

requester's quality preferences of the quality criteria and therefore do not consider the weight or priority of each quality criterion.

The quality service selection system (QSSS) is evaluated by comparing between selecting the best book from the Amazon E-Commerce Service (ECS) without using QSSS and selecting the best service from the ECS by using QSSS. It is seen that when sending a request to ECS, the requester can't easily select manually the best book because of the huge number of returned books. Whereas by using the QSSS the requester can automatically selects the best book based on his/her quality preferences.

Four scenarios are presented in order to evaluate the efficiency of the QSSS. It is noticed that the QSSS has a main drawback that it is only consider the monotonically increasing sub-criteria (e.g. Availability) and does not consider the monotonically decreasing (e.g., Service Price) when selecting the best service. Because of the time constrain, the drawback of the service selection will be addressed in the future work.

Chapter 8 Conclusion and Future Work

8.1 Conclusion

This thesis has made the following four contributions to Web services technologies:

1. **Definition of a classification of quality criteria**

The quality criteria classification is created in Chapter 3, which organizes the most important quality criteria into four groups: Performance, Failure probability, Trustworthiness and Cost. Each criteria group contains sub-criteria quality that holds the same characteristics. Performance criteria group contains the following sub-criteria: capacity, response time, throughput and execution time. Failure Probability criteria group contains of the following sub-criteria: availability, reliability, accessibility and scalability. Trustworthiness criteria group contains the following sub-criteria: security and reputation. Cost criteria group contains the following sub-criteria: service price and execution price.

The quality criteria classification captures the descriptions of quality criteria from requester's perspective as well from provider's perspective that are applicable to all Web services. The classification is generic as in the quality model in [90] that can be applicable in various domains and to meet different requester's demands. The classification is also flexible and extensible as in [87], in which the new criteria group and sub-criteria can be added without fundamentally altering the mathematical model and the service selection techniques that build on top of the classification.

The quality criteria classification in this thesis is similar to the quality classification in [86], [5] and [87] in that they classify the quality criteria into groups with different perspectives.

The quality classification in [86] includes three groups: performance, safety and cost. Performance contains response time and throughput, safety contains availability and reliability and cost contains the service cost. The quality classification in [5] organizes the most important quality-of-service (QoS) important to Web services into four groups: QoS related to runtime, transaction support, configuration management and cost and security. Runtime group contains the following aspects: scalability, capacity, performance, reliability, availability, robustness/flexibility, exception handling and accuracy. Transaction support related QoS contains integrity aspect. Configuration management and cost related QoS contains the following aspects: regulatory, supported standard, stability, cost and completeness. Security related QoS contains the following aspects: authentication, authorization, confidentiality, accountability, traceability and auditability, data encryption and non-repudiation.

The quality classification in [87] classifies the QoS parameters into the following groups: general, Internet service specific and task specific. General QoS parameters contain performance (throughput), performance (latency), reliability and cost. Internet service specific QoS parameters contain availability, security, accessibility and regulatory. Task specific QoS parameters contain task specific parameter.

The quality criteria classification is implemented using XML Spy editor in order to design Quality Criteria XML Schema as seen in Appendix A. The Quality Criteria Schema is accommodated in the WSDL as described in the coming contribution.

2. **Extension of the Web Services Description Language (WSDL) with the quality criteria classification**

WSDL is extended in Chapter 3 to accommodate the above quality criteria classification. This extension enables the service requester to express his/her

quality requirement when sending a request and the providers to express their quality specifications through publishing the services.

Because WSDL is an XML based language, the quality classification is implemented using XML Spy in order to design Quality Criteria XML Schema. The Quality Criteria XML Schema is augmented in the *Service Implementation Document* part of the WSDL as in [89], [131], by adding a new element <QualityCriteria> element in the <service> element.

3. **Development of a quality-based web services architecture**

A quality-based Web service architecture (QWSA) is developed in Chapter 4 (see Figure 4-1). The QWSA extends the current Web service architecture with quality server, because the current Web service architecture does not offer comprehensive quality of the Web service support. The quality server consists of four main components: quality manager, quality matchmaker, quality report analyzer, and quality database.

The main tasks of the quality server are the following

1. Enables the service providers to register their quality descriptions and store it in the quality database.
2. Matches the quality requirement specified by the service requester against the quality specifications of the advertised services.
3. Assists the requester to choose the best available service based on quality criteria.
4. Receives a requesters' quality report based on his/her judgments after consuming the selected services.

The task 2 and 3 are defined well in Chapter 5, but task 1 and 4 required further investigation.

The quality matchmaking component is the core component in the quality server that implements the quality matchmaking process (QMP) in order to select the best service as described in the coming section.

The related Web services architectures provide several techniques for enabling quality aspects in the current Web service architecture. A QoS broker in [94], [92], [86], [93] and [98] is used as a mediator between the service requester and service provider in order to select the best service based on quality aspects. An UX server in [95] architecture facilitates requesters to discover services with good qualities. The server sorts the service results according the QoS requirements and sends the result back to the requester. A QoS certifier in [5] extends the current Web service architecture in order to discover Web services by considering the functional and non-functional requirements.

However, the aforementioned techniques are not well defined and need more details for describing how their techniques select the best service.

4. **Development of a quality matchmaker component and quality matchmaking process**

The service selection based on quality criteria depends on the quality matchmaking process. Chapter 5 introduces a quality matchmaker component and the quality matchmaking process (QMP).

The quality matchmaker component in the quality server is the core component in the proposed QWSA and it is well defined in Chapter 5. The quality matchmaker consists of the following components: Interface matchmaking, quality criteria matchmaking and mathematical matchmaking.

The quality matchmaker introduces four algorithms or filters: interface matchmaking, quality criteria matchmaking, quality value constraints matchmaking, and mathematical matchmaking algorithm or filter. These four

algorithms or filters use the quality matchmaker components in order to implement their roles.

The matchmaking process implements the above four algorithms or filters in order to select the best Web service. The mathematical matchmaking algorithm is the most important step that is based on the mathematical model. Two techniques are used in the mathematical model:

1. Analytical Hierarchy Process (AHP) in order to calculate the criteria weights based on requester's preferences.
2. Euclidean distance which measures the distance between the requester's quality requirements and the providers' quality specifications. The Web service with the smallest distance is considered as the best service to select

The quality matchmaking process (QMP) is implemented in Chapter 6 by building a simulation program called a quality service selection system (QSSS). The QSSS program is developed by using C# Windows application in the Visual Studio .NET 2003 tool as a graphical user interface (GUI) to enable the service requester to specify his/her quality preferences. The QSSS program consists of the following forms and classes:

- Criteria Selection form
- Preference Selection form
- Sub-Criteria Selection form
- Sub-Preference Selection form
- Requirements Value form
- Utilities class

The functions of each form are described in Chapter 6. The hierarchy of the quality criteria group and the sub-criteria is based on the quality criteria classification that described in Chapter 3. The Utilities class consists of methods

which used to calculate the criteria and sub-criteria weight and to calculate the Euclidean distance between the quality requirement values specified by the service requester and the quality specifications offered by service providers.

The QMP process is evaluated by comparing it with the related approaches as described in Chapter 7. Also, the quality service selection system (QSSS) is evaluated in Chapter 7 by comparing between selecting the best book from the Amazon E-Commerce Service (ECS) without using QSSS and selecting the best service from the ECS by using QSSS. It is seen that when sending a request to ECS, the requester can't easily select manually the best book because of the huge number of returned books. Whereas by using the QSSS the requester can automatically selects the best book based on his/her quality preferences.

Four scenarios are presented in Chapter 7 in order to evaluate the efficiency of the QSSS. It is seen that the service requester can specify his/her quality requirements and preferences easily and select automatically the best Web service. But, it is noticed that the QSSS has a drawback that it is only consider the monotonically increasing sub-criteria (e.g. Availability) and does not consider the monotonically decreasing (e.g., Service Price) when selecting the best service. Because of the time constrain, the drawback of the service selection will be further investigated in the future work.

8.2 Future Work

Future enhancement of the proposed quality-based Web service architecture includes the following aspects:

1. Query type management

The proposed quality matchmaking process (QMP) has been derived with the assumption that the query, which is sent by the service requester, is volatile that is no new services will be added to UDDI and no changes to the quality criteria values for these services. These limitations will be further investigated by

adapting the requesters to any changes in the quality criteria during a long time query.

2. Notification mechanism

The functionality of the quality server needed to be extended with a notification mechanism to capture the dynamic nature of the quality criteria and sending a notification to quality manager of any changes in the quality criteria to keep update information in the quality database.

3. Feedback report

It is required a way to automate the collection of feedback report from the service requester after invoking the best service. The feedback report affects the final decision of service selection.

4. Quality specification publishing

This thesis has introduced quality-based service searching and selecting from the service requester side and not consider the quality-based service publishing from the service provider side. Hence, it is required a way to automate publishing of quality specifications from the service providers to the quality server.

5. Quality criteria type

The mathematical model considers only the increasing quality criteria such as Availability in the final decision of service selection. Further work needs to consider the decreasing quality criteria such as the Price. The proposed mathematical model normalise the performance matrix P, regarding the increasing quality criteria, with the equation $q_{ij} = \frac{p_{ij}}{\sqrt{\sum_{k=1}^{n} p^2{}_{ik}}}$,

It requires to use different equation to consider the decreasing criteria to normalise the performance matrix.0

6. Multi- queries management

Only one requester a time can query the quality-based web service architecture (QWSA) to select the best service. Further investigation needs to extend the functionality of the quality server to manage several queries that are sent concurrently by multi- requesters.

7. Quality criteria ontology

The quality criteria classification is a generic classification that can be applied in any domain. Further investigation needs to develop a quality criteria ontology that can be applied in a specific domain. The quality criteria ontology can be used to match services semantically and dynamically.

8. Quality matchmaking Process (QMP)

The QMP contains four algorithms: interface matchmaking algorithm, quality type matchmaking algorithm, quality value matchmaking algorithm and mathematical matchmaking algorithm. The interface matchmaking and the quality type matchmaking has demonstrated in Section 7.4.1.1. Where the requester sends a REST request to ECS database and retrieves the result of books. The result is saved in an Access database, which is further used in the implementation and evaluation of the quality matchmaking process (QMP). QMP is implemented by developing a quality service selection (QSSS) system. QMP implements the quality value matchmaking algorithm (Step-3) and the mathematical matchmaking algorithm (Step-4). Whereas, the interface matchmaking algorithm (Step-1) and the quality type matchmaking algorithm (Step-2) is already done and saved in the Access database. Further work need to implement Step-1 and Step-2 of the QMP in the QSSS system. It requires adding another window forms that enables the

requester to specify his/her functional requirements and match it with the functional specifications that are published in the UDDI registry.

.

References

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)," 4 February 2004. Available at: http://www.w3c.org/TR/REC-xml.

[2] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework," 24 June 2003. Available at :http://www.w3c.org/TR/SOAP12-part1.

[3] E. Christensen, F. Curbea, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," March 2001. Available at: http://www.w3.org/TR/wsdl.

[4] A. Manes, "Web Services Standardization: UDDI," 19 September 2003. Available at: http://www.uddi.org/news.html.

[5] S. Ran, "A Model for Web Services Discovery With QoS," *ACM SIGecom Exchanges*, vol. 4, pp. 1-10, 2003.

[6] H. M. Dietel, P. J. Dietel, B. Duwaldt, and L. K. Trees, *Web Services A Technical Introduction*: Prentice Hal, Upper Saddle River, 2003.

[7] H. Kreger, "Web Services Conceptual Architecture (WSCA 1.0),", IBM Software Group, May 2001.

[8] E. Newcomer, *Understanding Web Services*. Boston: Pearson Education, 2002.

[9] A. E. Walsh, *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book*: Prentice Hall, 2002.

[10] T. Bellwood, "Understanding UDDI," available at:http://www-128.ibm.com/developerworks/webservices/library/ws-featuddi/, 2002.

[11] "The UDDI Version 3 Specification," Available at: http://www.uddi.org, 2002.

[12] "The Universal Description, Discovery, and Integration (UDDI)," Available at: http://www.uddi.org. Last visited June 2004.

[13] L. Taher, H. Khatib, and R. Basha, "A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection," presented at The Second International Conference on Innovations in Information Technology (IIT'05), 2005.

[14] "Leveraging Web Services And 'Traditional' EAI.," April 2004. Available at: http://www.ebizq.net/hot_topics/web_services/features/4262.html?page=1. Last visited June 2004.

[15] J. W.-K. Hong, J.-S. Kim, and J.-K. Park, "A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications," *IEEE Network*, vol. 13, March/April 1999.

[16] M. Stal, "Web Services:Beyond Component-Based Computing," *Communications of the ACM*, vol. 45, pp. 71-76, October 2002.

[17] G. Glass, *Web Services: Building Blocks for Distributed Systems*: Prentice-Hall, 2002.

[18] P. Muschamp, "An Introduction to Web Services," *BT Technology Journal*, vol. 22, pp. 9-18, January 2004.

[19] E. Cerami, *Web Services Essentials*: O'Reilly & Associates, 2002.

[20] K. Gottshalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web Services Architecture," *IBM Systems Journal*, vol. 41, pp. 170-177, 2002.

[21] "Service-Oriented Architecture (SOA) Definition," Available at: http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html. last visited:June 2004.

[22] D. Barry, *Web Services and Service-Oriented Architecture*. San Francisco: Morgan Kaufmann Publishers, 2003.

[23] M. Colan, "Services-Oriented Architecture expands the Vision of Web Services, Part1," Available at:http://www6.software.ibm.com/software/developer/library/ws-soaintro.pdf . Last visited June 2004.

[24] W. Oellermann, *Architecting Web Services*: William L. Oellermann, Jr., 2001.

[25] M. Endrei, J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, D. M. Luo, and T. Newling, "Patterns: Service-oriented Architecture and Web Services," Redbook, SG24-6303-00, April 2004.

[26] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services Concepts, Architectures and Applications*: Springer-Verlag Berlin Heidelberg, 2004.

[27] W3C, "Web Services Architecture Requirements," Oct. 2002. Available at: http://www.w3.org/TR/wsa-reqs. last visited June 2004.

[28] Jupitermedia Corporation, "Webopedia:Online Dictionary for Computer and Internet Terms," Available at: http://www.webopedia.com.

[29] M. Serhani, "Web Services Roadmap," Concordia University, PhD Seminar January 2003.

[30] A. Ali, O. Rana, R. Al-Ali, and D. Walker, "UDDIe:An Extended Registry for Web Services," presented at Proceedings of the service oriented computing:Models,Architectures and Applications, SAINT-2003 IEEE Computer Society Press., Orlando Florida, USA., January 2003.

[31] R. Al-Ali, O. Rana, and D. Walker, "G-QoSM:Grid Service Discovery Using QoS Properties," presented at e-Science AHM02 Proceedings, Sheffield,UK, September 2002.

[32] S. Field and Y. Hoffner, "Web services and matchmaking," *International Journal of Networking and Virtual Organisations*, vol. 2, pp. 16-32, 2003.

[33] W.-T. Balke and M. Wagner, " Cooperative Discovery for User-centered Web Service Provisioning," presented at Proccedings of the International Conference on Web Services (ICWS'03), Las-Vegas, USA, 2003.

[34] I. W. S. A. Team, "Web Services Architecture Overview," 1 September 2000. Available at: http://www-106.ibm.com/developerworks/web/library/w-ovr/?dwzone=web . Last visited June 2004.

[35] The Microsoft .NET: http://www.microsoft.com/net.

[36] The SunONE: Available at: http://www.sun.com/sunone.

[37] J. Roy and A. Ramanujan, "Understanding Web Services," *IEEE IT Professional*, vol. 3, pp. 69-73, Nov./Dec. 2001.

[38] E. M. Maximilien and M. P. Singh "Toward autonomic web services trust and selection," presented at International Conference On Service Oriented Computing, New York, NY, USA, 2004.

[39] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web," *IEEE Internet Computing*, vol. 6, pp. 86-98, March/April 2002.

[40] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture.," 11 February 2004. Available at: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/wsa.pdf. Last visited June 2004.

[41] F. Leymann, "Web Services Flow Language (WSFL 1.0)," Available at: http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf. Last visited September 2004, 2001.

[42] A. Bergholz, "Extending Your Markup: An XML Tutorial," *IEEE Internet Computing*, vol. 4, pp. 74-79, July/August 2000.

[43] E. Wilde, "XML Technologies Dissected," *IEEE Internet Computing*, vol. 7, pp. 74-78, 2003.

[44] F. P. Coyle, *XML, Web Services and the Data Revolution*: Addison- Wesely, 2002.

[45] "Introduction to XML For Web Developers," Available at: http://www.extropia.com/tutorials/xml/history.html. Last visited June 2004.

[46] "The Simple Object Access Protocol (SOAP)," Available at :http://www.w3c.org/TR/SOAP12-part1. Last visited June 2004.

[47] Systinet, "Web Services: A practical Introduction to SOAP Web Services," White Paper, 2003.

[48] "Amazon E-Commerce Service Developer Guide," Available at: http://images.amazon.com/images/G/media/i3d/01/associates/aws-ecs-devguide_2005-01-19.pdf.

[49] R. Fielding, "Architectural Style and the Design of Network-based Software Architecture," PhD Thesis, University of California, Irvine.

Available at: http://www.ics.uci.edu/%7Efielding/pubs/dissertation/top.htm.

[50] R. Chinnici, M. Gudgin, J.-J. Moreau, J. Schlimmer, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0

Part 1: Core Language," Available at: http://www.w3.org/TR/2004/WD-wsdl20-20040803/wsdl20.pdf, 2004.

[51] T. A. Bellwood, "UDDI - A Foundation for Web Services," Available at: http://www.idealliance.org/papers/xml2001/papers/html/03-02-03.html . Last visited 16th August 2004.

[52] "UDDI Version 2.03 Data Structure Reference," 19 July 2002. Available at: http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm . Last visited July 2004.

[53] T. Bellwood, "UDDI Version 2.04 API Specification," Available at: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm, 2002.

[54] P. Brittenham and D. Ehnebuske, "Understanding WSDL in a UDDI Registry, Part1," Available at: http://www-106.ibm.com/developerworks/library/ws-wsdl/?n-ws-9201. Last visited March 2004, Sep 2002.

[55] F. Curbera, D. Ehnebuske, and D. Rogers, "Using WSDL in a UDDI Registry," May 21,2002.Available at: http://www.uddi.org/pubs/wsdlbestpractices-V1.07-Open-20020521.pdf. Last visited June 2004.

[56] Java 2 Platform Enterprise Edition (J2EE): http://java.sun.com/j2ee.

[57] J. Williams, "The Web services debate: J2EE vs. .NET," *Communications of the ACM*, vol. 46, pp. 58-63, 2003.

[58] C. Vawter and E. Roman, "J2EE vs. Microsoft.NET

A comparison of building XML-based web services," White Paper, 2001.

[59] Sun Microsystems, "The Java™ Architecture for XML Binding User's Guide," Available at: http://mediasrv.ns.ac.yu/extra/java2/specs/jaxb-docs.pdf, 2001.

[60] JAX-WS 2.0 specification: Available at: https://jax-ws.dev.java.net/.

[61] T. O'Donnell, "WSIT Components are Made Available," Available at: http://www.ftponline.com/special/javaart/interop/, 2006.

[62] XML and Web Services Security 3.0: Available at: https://xwss.dev.java.net/overview.html.

[63] Microsoft Corporation, *Developing XML Web Services and Server Components with Microsoft Visual Basic .NET and Microsoft Visual C# .NET*: Microsoft Press, 2003.

[64] S. Holzner, *Microsoft Visual C# .NET 2003*: Sams Publishing, 2003.

[65] J. J. Hanson, ".NET versus J2EE Web Services

A Comparison of Approaches," http://www.webservicesarchitect.com/content/articles/hanson01.asp, 2002.

[66] S. Graham, "The role of private UDDI nodes, Part 2: Private nodes and operator nodes," Web Services Architect, IBM Emerging Internet Technologies, 2001.

[67] S. Graham, "The role of private UDDI nodes in Web services, Part 1: Six species of UDDI," Web Services Architect, IBM Emerging Internet Technologies, 2001.

[68] North American Industry Classification System (NAICS): Available at: http://www.census.gov/epcd/www/naics.html, 2002.

[69] W.-T. Balke and M. Wagner, "Towards Personalized Selection of Web Services," presented at Proceedings of the International World Wide Web Conference (WWW2003). Budapest, Hungary, 2003.

[70] L. Li and I. Horrocks, "A Software Framework For Matching Based on Semantic Web Technology," presented at Proceedings International World Wide Web Conference (WWW2003), Budapest, Hungary., 2003.

[71] J. Colgrave, R. Akkiraju, and R. Goodwin, "External matching in UDDI," presented at IEEE International Conference on Web Services (ICWS'04), San Diego, California, June 2004.

[72] L. Taher, R. Basha, and H. Khatib, "QoS Information & Computation (QoS-IC) Framework for QoS-Based Discovery of Web Services," *UPGRADE*, vol. VI, 2005.

[73] C. Facciorusso, S. Field, R. Hauser, Y. Hoffner, R. Humbel, R. Pawlitzek, W. Rjaibi, and C. Siminitz, "A Web Services Matchmaking Engine for Web Services," presented at E-Commerce and Web Technologies: 4th International Conference, Czech, 2003.

[74] J. Hendler, T. Berners-Lee, and E. Miller, "Integrating Applications on the Semantic Web," *The Institute of Electrical Engineers of Japan*, vol. 122, pp. 676-680, 2002.

[75] Semantic Web Activity Group, "Semantic Web," Available at: http://www.w3.org/2001/sw/.

[76] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," presented at Proceedings of the International Semantic Web Conference (ISWC'02), Sardinia, Italy, 2002.

[77] M. Dean, D. Connolly, F. V. Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language 1.0 Reference," Available at: http://www.w3.org/TR/owl-ref, July 2002.

[78] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng, "DAML-S: Semantic Markup For Web Services," presented at Proceedings of the International Semantic Web Workshop, 2001.

[79] R. L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1048-1056, 1995.

[80] W. Stallings, *High-Speed Networks and Internets*, second ed: Prentice Hall, 2002.

[81] X. Xiao and L. M. Ni, "Internet QoS:A Big Picture," *IEEE Network*, vol. 13, pp. 8-18, March/April 1999.

[82] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services,", IBM DeveloperWorks Technical Paper, January 2002.

[83] D. A. Menasce', "QoS Issues in Web Services," *IEEE Internet Computing*, vol. 6, pp. 72-75, November-December 2002.

[84] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller, "A Concept for QoS Integration in Web Services," presented at 1st Web Services Quality Workshop (WQW 2003), Rome, 13th December 2003.

[85] M. Tian, A. Gramm, H. Ritter, and J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework," presented at IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), Beijing, China, 2004.

[86] Y.-J. Seo, H.-Y. Jeong, and Y.-J. Song, "A Study on Web Services Selection Method Based on the Negotiation Through Quality Broker: A MAUT-based Approach," presented at First International Conference on Embedded Software and Systems (ICESS 2004), Hangzhou, China, 2004.

[87] C. Patel, K. Supekar, and Y. Lee, "A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows," in *Database and Expert Systems Applications*, vol. 2736 / 2003: Springer-Verlag Heidelberg, October 2003, pp. 826 - 835.

[88] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Transactions on Software Engineering*, vol. 30, pp. 311 - 327, 2004.

[89] D. Gouscos, M. Kalikakis, and P. Georgiadis, "An Approach to Modeling Web Service QoS and Provision Price," presented at

4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Roma, Italy, December 13,2003.

[90] Y. Liu, A. H. Ngu, and L. Z. Zeng, "QoS computation and policing in dynamic web service selection," presented at International World Wide Web Conference, New York, NY, USA, 2004.

[91] M. Tian, A. Gramm, H. Ritter, and J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework."

[92] H. Chen, T. Yu, and K.-J. Lin, "QCWS: An Implementation of QoS-Capable Multimedia Web Services," presented at IEEE Fifth International Symposium on Multimedia Software Engineering (ISMSE'03), 2003.

[93] M. Serhani, A. Benharref, A. Hafid, and H. Sahraoui, "QoS Broker-Based Architecture for Web Services," Available at: http://www.iro.umontreal.ca/~serhanim/NOTERE04_finale_submission.pdf.

Last visited: June 2004.

[94] T. Yu and K.-J. Lin, "The Design of QoS Broker Algorithms for QoS-Capable Web Services," presented at IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE '04, Taipei, Taiwan, 2004.

[95] Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung, "UX-An Architecture Providing QoS-Aware and Federated Support for UDDI," presented at Proceeding of the first International Conference on Web Services (ICWS03), Las Vegas, Nevada, USA, 2003.

[96] D. A. Menasce', "QoS-Aware Software Components," *IEEE Internet Computing*, pp. 91-93, March/April2004.

[97] Alphaworks, "Web Services Toolkit," Available at:

http://www.alphaworks.ibm.com/tech/webservicestoolkit, 2003.

[98] P. Farkas and H. Charaf, "Web Services Planning Concepts," *Journal of WSCG*, vol. 11, pp. 3-7, February 2003.

[99] Y. Wang and E. Stroulia, "Flexible Interface Matching for Web-Services Discovery," presented at Proceedings of the 4th

International Conference on Wen Information Systems Engineering, 2003.

[100] K. Sycara, S. Widoff, M. Klusch, and J. Lu, "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 5, pp. 173-203, 2002.

[101] K. Sycara and M. Klusch, "Dynamic Service Matchmaking Among Agents in Open Information environments," *ACM SIGMOD Record (Special Isuue on Semantic Interoperability in Global Information Systems)*, vol. 28, pp. 47-53, 1999.

[102] M. Ouzzani and A. Bouguettaya, "Efficient Access to Web Services," *IEEE Internet Computing*, vol. 8, pp. 34-44, March/April 2004.

[103] Z. Chen, C. Liang-Tien, and L. Bu-Sung, "DAML-QoS Ontology for Web Services," presented at proceeding of the International Conference on Web Services 2004(ICWS04), San Diego, California, USA, 2004.

[104] T. Pilioura and A. Tsalgatidou, "PYRAMID-S:A Scalable Infrastructure for Semantic Web Service Publication," presented at 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications, Boston,Massachusetts, March 2004.

[105] P. Fedosseev, "Composition of Web Services and QoS Aspects," Seminar: Data Communication and Distributed Systems in the WS 2003/2004.

[106] C. Zhou, L.-T. Chia, and B.-S. Lee, "Semantics in Service Discovery and QoS Measurement," *IT Professional*, vol. 7, pp. 29 - 34, 2005.

[107] R. Sumra and A. D, "Quality of Service for Web Services-Demystification,Limitations, and Best Practice," Available at: http://www.developer.com/services/article.php/2027911. Last visited February 2004.

[108] E. M. Maximilien and M. P. Singh, "A Framework and Ontology for Dynamic Web Services Selection," *IEEE Internet Computing*, vol. 8, pp. 84-93, September-October 2004.

[109] ISO: Available at: http://www.iso.ch/iso/en/ISOOnline.frontpage.

[110] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality Driven Web services Composition," presented at Proceedings of the Twelfth International World Wide Web Conference (WWW'2003), Budapest,Hungary, May 2003.

[111] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, "QoS for Web Services: Requirements and Possible Approaches," 25 November 2003. Available at: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/ . Last visited July 2004.

[112] S. Rajesh and D. Arulazi, "Quality of Service for Web Services-Demystification,Limitations, and Best Practice," Available at: http://www.developer.com/services/article.php/2027911. Last visited February 2004.

[113] M. P. Singh, "Trustworthy Service Composition:

Challenges and Research Questions," *Lecture Notes on Artificial Intelligence*, vol. 2631, pp. 39-52, 2003.

[114] B. Atkinson, G. Della-Libera, S. Hada, and et al., "Web Services Security (WS-Security)," Available at: http://www-106.ibm.com/developerworks/webservices/library/ws-secure/, 2002.

[115] J. Zhang, "Trustworthy Web Services: Actions for Now," *IP Professional*, vol. 7, pp. 32-36, 2005.

[116] W. Chou, "Inside SSL: the secure sockets layer protocol," *IT Professional*, vol. 4, pp. 47-52, 2002.

[117] A. Karve, "Secure Communication over the Internet.," in *Network Magazine*, vol. 1, 1997.

[118] R. Salz, "Securing Web Services," Available at: http://webservices.xml.com/pub/a/ws/2003/01/15/ends.html, 2003.

[119] M. C. Mont, K. Harrison, and M. Sadler, "The HP time vault service: exploiting IBE for timed release of confidential information," presented at the 12th international conference on World Wide Web, Budapest, 2003.

[120] Reagle and Dillaway, "XML Encryption Syntax and Processing." Available at: http://www.w3.org/Encryption/2001/03/12-proposal.html, 2001.

[121] M. O'Neill, "We Know Web Services Need Security, But What Type?," *Web Services Journal*, vol. 3, pp. 18-21, 2003.

[122] M. Hondo, N. Nagaratnam, and A. Nadalin, "Securing Web Services," *IBM Systems Journal*, vol. 41, pp. 228-241, 2002.

[123] M. Chanliau, "The Security Challenge," *Web Services Journal*, vol. 3, pp. 32-36, 2003.

[124] M. Bartel, J. Boyer, B. Fox, and E. Simon, "XML-Signature Syntax and Processing." Available at: http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/, 2000.

[125] V. Tosic, K. Patel, and B. Pagurek, "WSOL - Web Service Offerings Language," presented at International Workshop on Web Services, E-Business, and the Semantic Web: CAiSE'02, Toronto, Canada, 2002.

[126] V. Tosic, B. Pagurek, B. Esfandiari, K. Patel, and W. Ma, "Web Service Offerings Language (WSOL) and Web Service Composition Management (WSCM)," presented at Workshop on Object- Oriented Web Services, Seattle, USA, 2002,.

[127] IBM Corporation, "Web Service Level Agreement (WSLA)

Language Specification," available at: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, 2003.

[128] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *Journal of Network and Systems Management*, vol. 11, March 2003.

[129] H. Ludwig, A. Keller, A. Dan, and R. P. King, " A Service Level Agreement Language for Dynamic Electronic Services," presented at 4th IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS'02), Newport Beach, USA, June, 2002.

[130] S. Andreozzi, D. Montesi, and R. Moretti, "Web Services Quality," presented at Conference on Computer,

Communication and Control Technologies (CCCT03), Orlando, 31 July - 2 August 2003.

[131] C. Marchetti, B. Pernici, and P. Plebani, "A Quality Model for e-Service Based Multi-Channel Adaptive Information Systems," presented at 4th International Conference on Web Information Systems Engineering Workshops (WISEW'03), Roma, Italy, December 13, 2003.

[132] XML Spy Home Edition: Available at: http://www.altova.com/support_freexmlspyhome.asp.

[133] Altova® Enterprise XML Suite 2006: Available at: http://www.altova.com/download_spy_enterprise.html.

[134] "Amazon Web Services," Available at: http://amazon.com/webservices.

[135] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Importing the Semantic Web in UDDI," presented at Proceedings of Web Services, E-Business and Semantic Web Workshop, CAiSE 2002., Toronto, Canada, 2002.

[136] A. Eleyan, L. Mikhailov, and L. Zhao, "Quality-of-Service Support in Web services Architecture," *ISI*, vol. 9, 2004.

[137] N. Thio and S. Karunasekera, "Automatic Measurement of a QoS Metric for Web Service Recommendation," presented at Software Engineering Conference, Australian, March 2005.

[138] T. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, vol. 15, pp. 234-281, 1977.

[139] T. L. Saaty, "How to make a decision: The Analytic Hierarchy Process," *European Journal of Operational Research*, vol. 48, pp. 9-26, 1990.

[140] M. Hajeeh and A. Al-Othman, "Application of the analytical hierarchy process in the selection of desalination plants," *Desalination*, vol. 174, pp. 97-108, 2005.

[141] E. W. L. Cheng and H. Li, "Information Priority-Setting for Better Resource Allocation Using Analytic Hierarchy Process (AHP)," *Information Management and Computer Security*, vol. 9, pp. 61-70, 2001.

[142] L. Taher, R. Basha, and H. El Khatib, "Establishing Association between QoS Properties in Service Oriented Architecture," presented at Proceedings of the IEEE International Conference on Next Generation Web Services Practices (NWeSP'05), 2005.

[143] L. Taher, H. El Khatib, and R. Basha, "A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection," presented at Second International Conference on Innovations in Information Technology (IIT'05), Dubai, UAE, 2005.

[144] M. Dunham, *Data Mining, Introductory and Advanced Topics*: Prentice Hall, upper Saddle River, New Jersy, 2003.

[145] H. Ye, B. Kerherve, and G. V. Bochmann, "QoS-based Distributed Query Processing," *Ingénierie des Systèmes d'Information (RSTI série ISI)*, vol. 9, 2004.

[146] P. Miseldine, "Use Amazon Web Services in ASP.NET," Available at: http://www.sitepoint.com/print/amazon-web-services-asp-net, 2004.

[147] S. Chandrasekaran, G. Silver, J. A. Miller, J. Cardoso, and A. P. Sheth, " Web service technologies and their synergy with simulation," presented at In Proceedings of the 2002 Winter Simulation Conference, 2002.

[148] S. Holzner, *Microsoft Visual C# .NET 2003*: Sams Publishing, 2004.

[149] Amazon E-Commerce Service: Available at: http://www.amazon.com/gp/browse.html/ref=sc_fe_l_2/102-0952141-0410546?%5Fencoding=UTF8&node=12738641&no=3435361&me=A36L942TSJ2AJA.

[150] "Architectural Style and the Design of Network-based Software Architecture," PhD Thesis, University of California, Irvine.

Available at: http://www.ics.uci.edu/%7Efielding/pubs/dissertation/top.htm.

[151] Visual Studio .NET: Available at: http://msdn2.microsoft.com/en-us/library/ms269115(vs.80).aspx.

Appendix A Quality Criteria XML Schema

Quality Criteria XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Amna Eleyan
(University of Manchester) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="QualityCriteria">
  <xs:annotation>
    <xs:documentation> root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Performance" type="PerformanceType"
                          minOccurs="0"/>
      <xs:element name="FailureProbability" type="FailureProbabilityType"
                          minOccurs="0"/>
      <xs:element name="Trustworthiness" type="TrustworthinessType"
                          minOccurs="0"/>
      <xs:element name="Cost" type="CostType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="qValueType">
  <xs:sequence>
    <xs:element name="Min" type="xs:double"/>
    <xs:element name="Max" type="xs:double"/>
    <xs:element name="Preferred" type="xs:double"/>
  </xs:sequence>
  <xs:attribute name="qlevel" type="qlevelType" use="required"/>
</xs:complexType>
<xs:simpleType name="unitType">
  <xs:restriction base="xs:string">
    <xs:enumeration value=""/>
    <xs:enumeration value="Msec"/>
    <xs:enumeration value="Request/sec"/>
    <xs:enumeration value="Percentage"/>
    <xs:enumeration value="Pound"/>
    <xs:enumeration value="None"/>
  </xs:restriction>
</xs:simpleType>
```

Appendix A Quality Criteria XML Schema

```
<xs:simpleType name="qlevelType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="High"/>
    <xs:enumeration value="Medium"/>
    <xs:enumeration value="Low"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="weightType">
  <xs:restriction base="xs:double">
    <xs:maxInclusive value="1"/>
    <xs:minInclusive value="0"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="qCriteriaType">
  <xs:sequence>
    <xs:element name="qValue">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="qValueType">
            <xs:attribute/>
            <xs:anyAttribute/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="unit" type="unitType"/>
    <xs:element name="weight" type="weightType"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="PerformanceType">
  <xs:complexContent>
    <xs:extension base="subCriteriaType">
      <xs:sequence>
        <xs:element name="Capacity" type="qCriteriaType"
                                    minOccurs="0"/>
                                    <xs:element name="ResponseTime"
                                    type="qCriteriaType" minOccurs="0"/>
                                    <xs:element name="Latency" type="qCriteriaType"
                                    minOccurs="0"/>
                                    <xs:element name="Throughput"
                                    type="qCriteriaType" minOccurs="0"/>
                                    <xs:element name="ExecutionTime"
                                    type="qCriteriaType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="FailureProbabilityType">
  <xs:sequence>
    <xs:element name="Availability" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="Reliability" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="Accessibility" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="Accuracy" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="Scalability" type="qCriteriaType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="TrustworthinessType">
  <xs:sequence>
    <xs:element name="Reputation" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="Security" type="qCriteriaType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="CostType">
  <xs:sequence>
    <xs:element name="ServicePrice" type="qCriteriaType" minOccurs="0"/>
    <xs:element name="TransactionPrice" type="qCriteriaType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

Appendix B: Quality Service Selection System

This appendix displays the source codes of quality service selection system, which called *QSSS* using C# Windows application implemented using Visual Studio .NET 2003. QSSS consists of several forms in order to enable the service requester to specify his quality preferences and sub-criteria quality levels. Quality criteria classification and mathematical model are used to assist the service requester to select the best candidates Web service. These forms and techniques are explained in the following sections.

B-1 CriteriaSelection Form

Figure B-1 shows the Criteria Selection form.

Figure B-1 CriteriaSelection Form

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
```

Appendix B: Quality Service Selection System

```
namespace ServiceSelection2
{
public class CriteriaSelection : System.Windows.Forms.Form
{
  private System.Windows.Forms.Label label1;
  private System.Windows.Forms.Label label2;
  private System.Windows.Forms.Button Next;
  private System.Windows.Forms.Button Exit;
  public System.Windows.Forms.CheckBox checkBox1;
  public System.Windows.Forms.CheckBox checkBox2;
  public System.Windows.Forms.CheckBox checkBox3;
  public System.Windows.Forms.CheckBox checkBox4;
  Utilities utility=new Utilities();
  private double [] oneValue=new double[1];
        public static double[] weightCriteria1=new double[1];
  static int MAX_NUM = 4;
  public static CheckBox []checkBoxesArray = new
        CheckBox[MAX_NUM];
  public CriteriaSelection()
  {
    // Required for Windows Form Designer support
    InitializeComponent();
    oneValue[0]=1.0;
  }
  static void Main()
  {
  Application.Run(new CriteriaSelection());
  }

  private void Exit_Click(object sender, System.EventArgs e)
  {
  if(MessageBox.Show("Are You Sure You want to exit", "message
        Box", MessageBoxButtons.OKCancel)==DialogResult.OK)
    {
    base.Dispose();
    }
  }

// count number of quality criteria selected by service requester
  static public int numOfCriteria;
  private void updateNumOfCriteria(){
    numOfCriteria=0;
    if (checkBox1.Checked) numOfCriteria++;
    if (checkBox2.Checked) numOfCriteria++;
    if (checkBox3.Checked) numOfCriteria++;
```

```
    if (checkBox4.Checked) numOfCriteria++;
  }

  private void Next_Click(object sender, System.EventArgs e)
  {
     updateNumOfCriteria();
     SubCriteriaSelection subForm1=new SubCriteriaSelection();
  preferenceSelection form = new preferenceSelection();
  CriteriaSelection CriteriaSelection=new CriteriaSelection();
//create M matrix instance from Matrix class
     Matrix M=new Matrix(numOfCriteria, numOfCriteria);

//requester has to select at least one quality criteria
  if(checkBox1.Checked==false && checkBox2.Checked==false &&
     checkBox3.Checked==false && checkBox4.Checked==false)
  {
  MessageBox.Show("Please make sure you select at least one
     criteria ", "message Box", MessageBoxButtons.OK);
  }

//if only Performance criterion is selected
  else if (checkBox4.Checked==false checkBox2.Checked==false&&
     checkBox3.Checked==false && checkBox1.Checked==true)
  {
  utility.fillMatrix0(M,oneValue);
//calculate the weight of selected criteria by calling //calculateWeights method
from Utilities class
  weightCriteria1=utility.calculateWeights(M,numOfCriteria);
  subForm1.groupBox1.Enabled=true;// switch to SubCriteriaSelection form
  subForm1.Show();
  }
//if Failure Probability and Trustworthiness are selected
  else if(checkBox2.Checked==true &&
     checkBox3.Checked==true&&checkBox4.Checked==false &&
     checkBox1.Checked==false)
    {
    form.comboBox1.Visible=false;
    form.comboBox2.Visible=false;
    form.comboBox3.Visible=false;
    form.comboBox5.Visible=false;
    form.comboBox6.Visible=false;
    form.PFlabel.Visible=false;
    form.PTlabel.Visible=false;
    form.PClabel.Visible=false;
    form.FClabel.Visible=false;
```

```
    form.TClabel.Visible=false;
    form.FTlabel.Location = new
          System.Drawing.Point(0, 65);
    form.comboBox4.Location=new
          System.Drawing.Point(300, 65);
    checkBoxesArray[0] = checkBox2;
    checkBoxesArray[1] = checkBox3;
    form.Show();// open preferenceSelection form
    }

//if Failure Probability, Trustworthiness and Cost are selected
  else if(checkBox2.Checked==true && checkBox3.Checked==true
     && checkBox4.Checked==true&& checkBox1.Checked==false)
  {
    form.comboBox1.Visible=false;
    form.comboBox2.Visible=false;
    form.comboBox3.Visible=false;
    form.PFlabel.Visible=false;
    form.PTlabel.Visible=false;
    form.PClabel.Visible=false;
    form.FTlabel.Location = new System.Drawing.Point(0,
          65);
    form.FClabel.Location = new System.Drawing.Point(0,
          90);
    form.TClabel.Location = new System.Drawing.Point(0,
          115);
    form.comboBox4.Location=new System.Drawing.Point(300,
          65);
    form.comboBox5.Location=new System.Drawing.Point(300,
          90);
    form.comboBox6.Location=new System.Drawing.Point(300,
          115);
    checkBoxesArray[0] = checkBox2;
    checkBoxesArray[1] = checkBox3;
    checkBoxesArray[2] = checkBox4;
    form.Show();// open preferenceSelection form
    }

//if Performance, Failure Probability, Trustworthiness and Cost //are selected
  else if(checkBox1.Checked==true && checkBox2.Checked==true&&
     checkBox3.Checked==true && checkBox4.Checked==true)
  {
  checkBoxesArray[0] = checkBox1;
    checkBoxesArray[1] = checkBox2;
    checkBoxesArray[2] = checkBox3;
```

```
    checkBoxesArray[3] = checkBox4;
    form.Show();// open preferenceSelection form
  }}   }
```

B-2 PreferenceSelection Form

Figure B-2 shows the PreferenceSelection Form

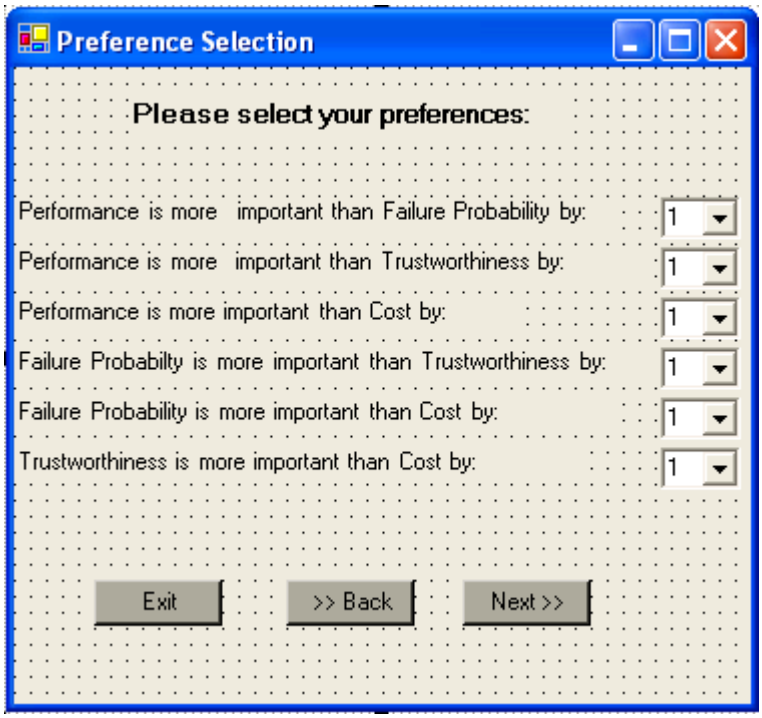

Figure B-2 PreferenceSelection Form

```
namespace ServiceSelection2
{
  public class preferenceSelection : System.Windows.Forms.Form
  {
  public System.Windows.Forms.Label label1;
  private System.Windows.Forms.Button Next;
  private System.Windows.Forms.Button Back;
  private System.Windows.Forms.Button Exit;
  public System.Windows.Forms.ComboBox comboBox1;
  public System.Windows.Forms.ComboBox comboBox2;
  public System.Windows.Forms.ComboBox comboBox3;
  public System.Windows.Forms.ComboBox comboBox4;
  public System.Windows.Forms.ComboBox comboBox5;
  public System.Windows.Forms.ComboBox comboBox6;
  public System.Windows.Forms.Label PTlabel;
  public System.Windows.Forms.Label FTlabel;
```

```
  public System.Windows.Forms.Label PClabel;
  public System.Windows.Forms.Label FClabel;
  public System.Windows.Forms.Label TClabel;
  public System.Windows.Forms.Label PFlabel;
     SubCriteriaSelection subform=new SubCriteriaSelection();
     public static double[] weightCriteria=new double[4];
  Utilities utility=new Utilities();
  static int MAX_NUM_OF_VALUES = 6;
  double [] valuesArray = new double[MAX_NUM_OF_VALUES];
  ComboBox [] comboBoxesArray = new
     ComboBox[MAX_NUM_OF_VALUES];
  double consistency;
  public preferenceSelection()
  {
  // Required for Windows Form Designer support
    InitializeComponent();
    comboBoxesArray[0] = comboBox1;
    comboBoxesArray[1] = comboBox2;
    comboBoxesArray[2] = comboBox3;
    comboBoxesArray[3] = comboBox4;
    comboBoxesArray[4] = comboBox5;
    comboBoxesArray[5] = comboBox6;
    }
      }

private void Exit_Click(object sender, System.EventArgs e)
  {
  if(MessageBox.Show("Are You Sure You want to exit", "message
     Box", MessageBoxButtons.OKCancel)==DialogResult.OK)
    {
    base.Dispose();
    }
  }
  private void Back_Click(object sender, System.EventArgs e)
  {
    this.Close();
  }

//This method opens SubCriteriaSelection form which based on the
// selected criteria group if CriteriaSelection form
  public static void subFormEnable()
  {
  for(int i=0; i< CriteriaSelection.allBoxesArray.Length; i++)
  {
  if(CriteriaSelection.allBoxesArray[i].Checked)
```

```
    {
    SubCriteriaSelection.groupBoxesArray[i].Enabled=true;
    }
  }
  }

  private void Next_Click(object sender, System.EventArgs
     e)
  {
    Result result=new Result();
    Result result1=new Result();
    Matrix A=new Matrix(CriteriaSelection.numOfCriteria,
          CriteriaSelection.numOfCriteria);

// Convert comboBox1,2,3,4,5,6 from text values to double values
  for (int i =0; i< comboBoxesArray.Length; i++)
    {
    if(comboBoxesArray[i].Visible==true)
      {
      utility.convert(i, comboBoxesArray[i],
              valuesArray);
      }
    }
//construct pair-wise comparison matrix A by filling it with //requester's
preference values by calling fillMatrix0() method //from Utilities class
  utility.fillMatrix0(A, valuesArray);

// calculate the criteria weight by calling calculateWeights() //method from
Utilities class
weightCriteria = utility.calculateWeights(A,CriteriaSelection.numOfCriteria);

//calculate the Consisteny Ratio (CR)if the number of
//selected quality criteria is more than two by calling //ConsistencyRatio()method
from Utilities class

if(CriteriaSelection.numOfCriteria>2 && CriteriaSelection.numOfCriteria<=10)
  {
  consistency=utility.ConsistencyRatio(A, weightCriteria,
     CriteriaSelection.numOfCriteria);

  }
//if the Consistency Ratio(CR) is less than 0.1 then the judgement is consistent
and the requester can continue the selection process

if(consistency<0.1)
```

```
  {
    subFormEnable();
    subform.Show();    }
  else
// if the Consistency Ratio (CR) is more than 0.1 then the requester has to specify
new preferences values
MessageBox.Show("Please enter new quality preferences values ", "message
Box", MessageBoxButtons.OK);
  }   }}
```

B-3 SubCriteriaSelection Form

Figure B-3 shows the *SubCriteriaSelection* form.

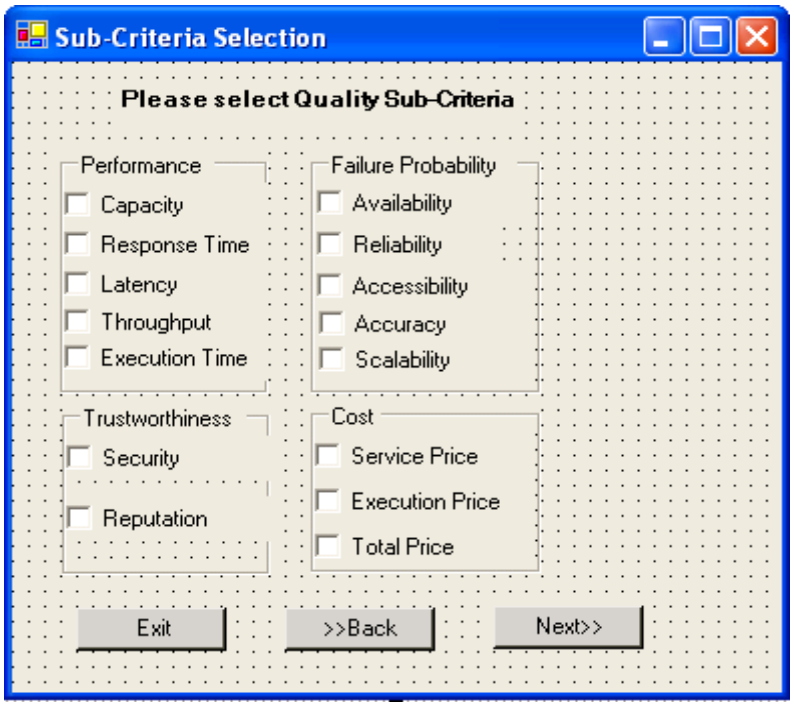

Figure B-3 SubCriteriaSelection

```
public class SubCriteriaSelection : System.Windows.Forms.Form
  {
  public System.Windows.Forms.GroupBox groupBox4;
        public System.Windows.Forms.GroupBox groupBox3;
  public System.Windows.Forms.GroupBox groupBox2;
  public System.Windows.Forms.GroupBox groupBox1;
        public System.Windows.Forms.CheckBox respBox;
        public System.Windows.Forms.CheckBox thptBox;
        public System.Windows.Forms.CheckBox avalBox;
        public System.Windows.Forms.CheckBox relBox;
```

```
        public System.Windows.Forms.CheckBox secBox;
        public System.Windows.Forms.CheckBox repBox;
        public System.Windows.Forms.CheckBox serpBox;
        public System.Windows.Forms.CheckBox expBox;
        private System.Windows.Forms.Button Exit;
  private System.Windows.Forms.Button Back;
        private System.Windows.Forms.Button Next;
      public static int  totalSubNum;
  Utilities utility=new Utilities();
  private double [] subValue=new double[1];
  public double[] weightSubCriteria=new double[1];
     public double[] weightSubCriteria1=new double[1];
     public double[] weightSubCriteria2=new double[1];
     public double[] weightSubCriteria3=new double[1];
     public static double[] totalWeight=new double[4];
        public SubCriteriaSelection()
  {
// Required for Windows Form Designer support
    InitializeComponent();
    subValue[0]=1;

private void Exit_Click(object sender, System.EventArgs e)
  {
  if(MessageBox.Show("Are You Sure You want to exit", "message Box",
            MessageBoxButtons.OKCancel)==DialogResult.OK)
    {
    base.Dispose();
    }

     }

private void Back_Click(object sender, System.EventArgs e)
  {
    this.Close();
  }
  public static int numOfSubCriteria;
  public static int numOfSubCriteria1;
  public static int numOfSubCriteria2;
  public static int numOfSubCriteria3;
// Count the number of sub-criteria selected in each criteria group(Performance,
// Failure Probability, Trustworthiness and Cost)
  private void updateNumOfSubCriteria()
  {
    numOfSubCriteria=0;
    numOfSubCriteria1=0;
    numOfSubCriteria2=0;
```

```
    numOfSubCriteria3=0;

    if (respBox.Checked==true) numOfSubCriteria++;
    if (thptBox.Checked==true) numOfSubCriteria++;

    if (avalBox.Checked==true) numOfSubCriteria1++;
    if (relBox.Checked==true) numOfSubCriteria1++;

    if (secBox.Checked==true) numOfSubCriteria2++;
    if (repBox.Checked==true) numOfSubCriteria2++;

    if (serpBox.Checked==true) numOfSubCriteria3++;
    if (expBox.Checked==true) numOfSubCriteria3++;
     }

private void Next_Click(object sender, System.EventArgs e)
  {
    RequirementsValue requirement=new RequirementsValue();
    SubPreferenceSelection subprefSelection=new SubPreferenceSelection();
    updateNumOfSubCriteria();
    Matrix P=new Matrix(numOfSubCriteria,numOfSubCriteria);
    Matrix P1=new Matrix(numOfSubCriteria1,numOfSubCriteria1);
    Matrix P2=new Matrix(numOfSubCriteria2,numOfSubCriteria2);
    Matrix P3=new Matrix(numOfSubCriteria3,numOfSubCriteria3);
// Requester has to select at least one sub-criteria in each criteria group
if((groupBox1.Enabled==true &&
respBox.Checked==false&&thptBox.Checked==false)||(groupBox2.Enabled==tru
e&&avalBox.Checked==false&&relBox.Checked==false)||(groupBox3.Enabled=
=true && repBox.Checked==false&&secBox.Checked==false)||
(groupBox4.Enabled==true &&serpBox.Checked==false&&
expBox.Checked==false))
  {
  MessageBox.Show("Please make sure you select at least one Sub-criteria from
each criteria group", "message Box", MessageBoxButtons.OK);
  }

 //If only one quality sub-criterion is selected within each quality criteria group
//then this form will jump to select requirement values from RequirementsValue
form

//and skip SubPreferenceSelection form.

//Select only Availability
if(respBox.Checked==false&&
thptBox.Checked==false&&avalBox.Checked==true&&relBox.Checked==false
&&
```

```
secBox.Checked==false&&repBox.Checked==false&&serpBox.Checked==false
&&expBox.Checked==false)
{
            totalSubNum=numOfSubCriteria1;
  result.textBox1.AppendText("The number of criteria : "+numOfSubCriteria1);
  utility.fillMatrix0(P1,subValue);
//calculate the weight of sub-criteria in each criteria group by calling
calculateWeights method
  weightSubCriteria1=utility.calculateWeights(P1, numOfSubCriteria1);
  requirement.label10.Visible=false;
  requirement.label5.Visible=false;
  requirement.label6.Visible=false;
  requirement.label7.Visible=false;
  requirement.label3.Visible=false;
  requirement.label9.Visible=false;
  requirement.label8.Visible=false;
  requirement.comboBox1.Visible=false;
  requirement.comboBox5.Visible=false;
  requirement.comboBox6.Visible=false;
  requirement.comboBox7.Visible=false;
  requirement.comboBox3.Visible=false;
  requirement.comboBox4.Visible=false;
  requirement.comboBox8.Visible=false;
  requirement.label2.Location=new System.Drawing.Point(0, 40);
  requirement.comboBox2.Location=new System.Drawing.Point(242, 40);
// The final weight array "totalWeight" which is used in the Euclidean distance
calculation
  for(int j=0; j<totalSubNum;j++)
  {
  totalWeight[j]=Math.Round(weightSubCriteria1[j]*CriteriaSelection.weightCrit
eria1[j],3
            );
  requirement.Show();//open RequirementsValue form
  }

//If two sub-criteria are selected in each criteria group then the requester needs to
select their //preferences or importance

//Select Availability and Reliability
if(respBox.Checked==false&&
thptBox.Checked==false&&avalBox.Checked==true&&relBox.Checked==true&
&
secBox.Checked==false&&repBox.Checked==false&&serpBox.Checked==false
&&expBox.Checked==false)
{
```

```
            totalSubNum=numOfSubCriteria1;
          subprefSelection.comboBox1.Visible=false;
  subprefSelection.comboBox3.Visible=false;
  subprefSelection.comboBox4.Visible=false;
  subprefSelection.label2.Visible=false;
  subprefSelection.label4.Visible=false;
  subprefSelection.label5.Visible=false;
  subprefSelection.comboBox2.Location=new System.Drawing.Point(323, 65);
  subprefSelection.label3.Location = new System.Drawing.Point(0, 65);
  subprefSelection.Show();  //open SubPreferenceSelection form
  }

//Select Availability ,Reputation and Service Price
if(respBox.Checked==false&&thptBox.Checked==false&&avalBox.Checked==tr
ue&&relBox.Checked==false&&secBox.Checked==false&&repBox.Checked==t
rue&&serpBox.Checked==true&&expBox.Checked==false)
{

totalSubNum=numOfSubCriteria1+numOfSubCriteria2+numOfSubCriteria3;
       utility.fillMatrix0(P1,subValue);
       utility.fillMatrix0(P2,subValue);
       utility.fillMatrix0(P3,subValue);
//calculate the weight of sub-criteria in each criteria group by calling
/calculateWeights() method from Utilities class
        weightSubCriteria1=utility.calculateWeights(P1, numOfSubCriteria1);
        weightSubCriteria2=utility.calculateWeights(P2, numOfSubCriteria2);
         weightSubCriteria3=utility.calculateWeights(P3, numOfSubCriteria3);
         double [] weightSubArray=new double[totalSubNum];
         weightSubCriteria1.CopyTo(weightSubArray,0);
         weightSubCriteria2.CopyTo(weightSubArray,1);
         weightSubCriteria3.CopyTo(weightSubArray,2);

// The final weight array "totalWeight" which is used in the Euclidean distance
calculation
for(int k=0; k<totalSubNum;k++)
{
totalWeight[k]=Math.Round(weightSubArray[k]*preferenceSelection.weightCrite
ria[k],3
            );
  }
  requirement.label10.Visible=false;
           requirement.label5.Visible=false;
  requirement.label6.Visible=false;
  requirement.label7.Visible=false;
  requirement.label8.Visible=false;
```

```
  requirement.comboBox1.Visible=false;
            requirement.comboBox5.Visible=false;
  requirement.comboBox6.Visible=false;
  requirement.comboBox7.Visible=false;
  requirement.comboBox8.Visible=false;
  requirement.label2.Location=new System.Drawing.Point(0, 10);
  requirement.comboBox2.Location=new System.Drawing.Point(242, 10);
  requirement.label3.Location=new System.Drawing.Point(0, 40);
  requirement.comboBox3.Location=new System.Drawing.Point(242, 40);
  requirement.label9.Location=new System.Drawing.Point(0, 70);
  requirement.comboBox4.Location=new System.Drawing.Point(242, 70);
  requirement.Show();//open RequirementsValue form


  }

//Select Response time,Throughput, Availability, Reliability, Security, Reputation,
Service Price and Execution Price
if(respBox.Checked==true&&
thptBox.Checked==true&&avalBox.Checked==true&&
relBox.Checked==true&&secBox.Checked==true&&repBox.Checked==true&&s
erpBox.Checked==true&&expBox.Checked==true)
{
          totalSubNum=numOfSubCriteria+numOfSubCriteria1+numOfSubCriteria
2+numOfSubC
           riteria3;
  subprefSelection.Show();  //open SubPreferenceSelection form }
```

B-4 SubPreference Selection Form

Figure B-4 shows the SubPreferenceSelection Form

Appendix B: Quality Service Selection System

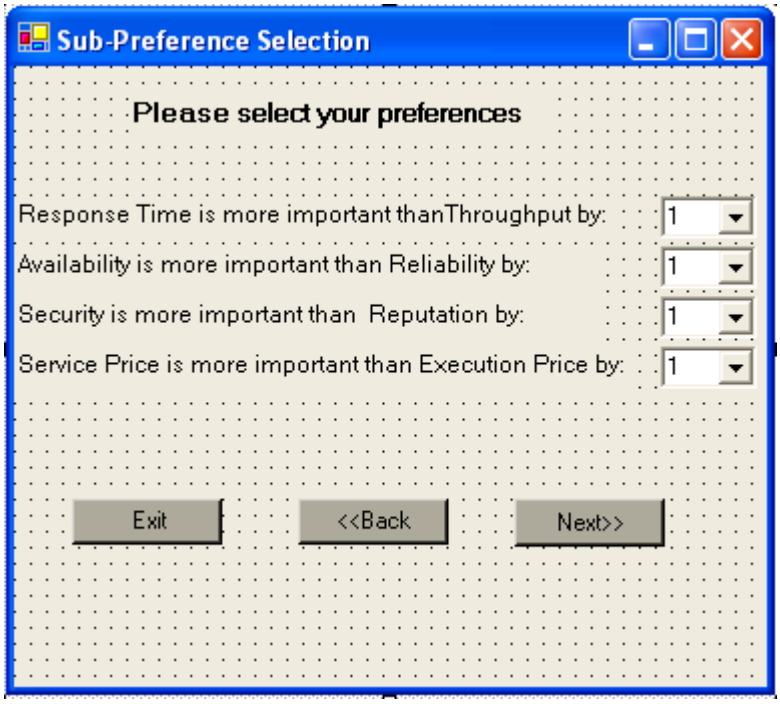

Figure B-4 SubPreferenceSelection Form

```
public class SubPreferenceSelection : System.Windows.Forms.Form
  {
  public System.Windows.Forms.ComboBox comboBox1;
  public System.Windows.Forms.ComboBox comboBox2;
  public System.Windows.Forms.ComboBox comboBox3;
  public System.Windows.Forms.ComboBox comboBox4;
  private System.Windows.Forms.Button Exit;
  private System.Windows.Forms.Button Back;
    private System.Windows.Forms.Button Next;
  public static int  subCriteriaNum;
  public  static double[] totalWeight=new double[8];
  private double []weightSub=new double[2];
    private  double[] totalWeightP=new double[2];
    private  double[] totalWeightFP=new double[2];
    Utilities utility1=new Utilities();
    static int MAX_NUM_OF_VALUES=4;
    double [] subValuesArray = new double[MAX_NUM_OF_VALUES];
    ComboBox [] subComboArray = new
ComboBox[MAX_NUM_OF_VALUES];
     public SubPreferenceSelection()
  {// Required for Windows Form Designer support
    InitializeComponent();
    subComboArray[0] = comboBox1;
    subComboArray[1] = comboBox2;
    subComboArray[2] = comboBox3;
```

```
      subComboArray[3] = comboBox4;

private void Back_Click(object sender, System.EventArgs e){
this.Close();}
private void Exit_Click(object sender, System.EventArgs e){
base.Dispose(); }
//Calculate Performance' sub-criteria weight
private double[] TotalWeightPerformance(
{
    Matrix B=new Matrix(SubCriteriaSelection.numOfSubCriteria,
     SubCriteriaSelection.numOfSubCriteria);
    for (int i =0; i< subComboArray.Length; i++)
    {
      if(subComboArray[i].Visible==true)
      {
      utility1.convert(i, subComboArray[i], subValuesArray);
      } }
//Construct pair-wise comparison matrix regarding Performance' sub-criteria
//preferences by calling fillMatrix0() method from Utilities class
  utility1.fillMatrix0(B, subValuesArray);
  double[] weightSCriteria =
  utility1.calculateWeights(B,SubCriteriaSelection.numOfSubCriteria);
//Performance weight "totalWeightP" calculation if more than
//one criteria group are selected
  if(CriteriaSelection.numOfCriteria>1)
  { for(int k=0; k<SubCriteriaSelection.numOfSubCriteria;k++)
    {
totalWeightP[k]=Math.Round(preferenceSelection.weightCriteria[k]*weightSCrit
eria[k],3);
      }  }
//Performance weight "totalWeightP" calculation if only
// Performance criteria group is selected
  else if(CriteriaSelection.numOfCriteria==1)
  {
  CriteriaSelection.weightCriteria1.CopyTo(weightSub,0);
  CriteriaSelection.weightCriteria1.CopyTo(weightSub,1);
  for(int k=0; k<SubCriteriaSelection.numOfSubCriteria;k++)
    {
          totalWeightP[k]=Math.Round(weightSub[k]*weightSCriteria[k],3);
    }
  }
   return totalWeightP;
  }
    //Calculate Failure Probability' sub-criteria weight
  private double[] TotalWeightFP()
```

```
  {
   Result result1=new Result();
   Matrix B=new Matrix(SubCriteriaSelection.numOfSubCriteria1,
                SubCriteriaSelection.numOfSubCriteria1);
   for (int i =0; i< subComboArray.Length; i++)
   {
    if(subComboArray[i].Visible==true)
    {
    utility1.convert(i, subComboArray[i], subValuesArray);
    }
   }
//Construct pair-wise comparison matrix regarding Failure Probability ' sub-
criteria preferences by calling fillMatrix0() method from Utilities class
  utility1.fillMatrix0(B, subValuesArray);
        double[] weightSCriteria =
        utility1.calculateWeights(B,SubCriteriaSelection.numOfSubCriteria1);
/Failure Probability weight "totalWeightFP" calculation if more than
//one criteria group are selected
  if(CriteriaSelection.numOfCriteria>1)
  {
  for(int k=0; k<SubCriteriaSelection.numOfSubCriteria1;k++)
  {
  totalWeightFP[k]=Math.Round(preferenceSelection.weightCriteria[k]*wei
        ghtSCriteria[k],3);
  }
  }
//Failure Probability weight "totalWeightFP" calculation if only
// Failure Probability criterion group is selected
  else if(CriteriaSelection.numOfCriteria==1)
  {
  CriteriaSelection.weightCriteria1.CopyTo(weightSub,0);
  CriteriaSelection.weightCriteria1.CopyTo(weightSub,1);
  for(int k=0; k<SubCriteriaSelection.numOfSubCriteria1;k++)
  {
  totalWeightFP[k]=Math.Round(weightSub[k]*weightSCriteria[k],3);
  }
  }
        return totalWeightFP;
    }

private void Next_Click(object sender, System.EventArgs e)
  {
  RequirementsValue requirementValue=new RequirementsValue();
  SubCriteriaSelection subSelection=new SubCriteriaSelection();
// if Performance and Failure Probability criteria group are selected
```

```
        if(label2.Visible==true&&label3.Visible==true&&label4.Visible==false&
        & label5.Visible==false)
  {
    requirementValue.label3.Visible=false;
    requirementValue.label7.Visible=false;
    requirementValue.label8.Visible=false;
    requirementValue.label9.Visible=false;
    requirementValue.comboBox3.Visible=false;
    requirementValue.comboBox4.Visible=false;
    requirementValue.comboBox7.Visible=false;
    requirementValue.comboBox8.Visible=false;
    requirementValue.label5.Location=new System.Drawing.Point(0,
                    40);
    requirementValue.comboBox5.Location=new
                    System.Drawing.Point(242, 40);
    requirementValue.label2.Location=new System.Drawing.Point(0,
                    70);
    requirementValue.comboBox2.Location=new
                     System.Drawing.Point(242, 70);
    requirementValue.label6.Location=new System.Drawing.Point(0,
                     100);
    requirementValue.comboBox6.Location=new
                    System.Drawing.Point(242, 100);
  subCriteriaNum=SubCriteriaSelection.numOfSubCriteria+SubCriteriaSele
        ction.numOfSubCriteria1;
        TotalWeightPerformance();// calculate Performance' sub-criteria weight
  TotalWeightFP();
// calculate Failure probabilty' sub-criteria  weight
 //the final weight "totalWeight" will be used in Euclidean distance calculation

  totalWeightP.CopyTo(totalWeight,0);
  totalWeightFP.CopyTo(totalWeight,2);
     }

        requirementValue.Show();//open RequirementsValue form
      }
```

B-5 RequirementsValue Form

Figure B-5 shows the *RequirementsValue* Form.

Appendix B: Quality Service Selection System

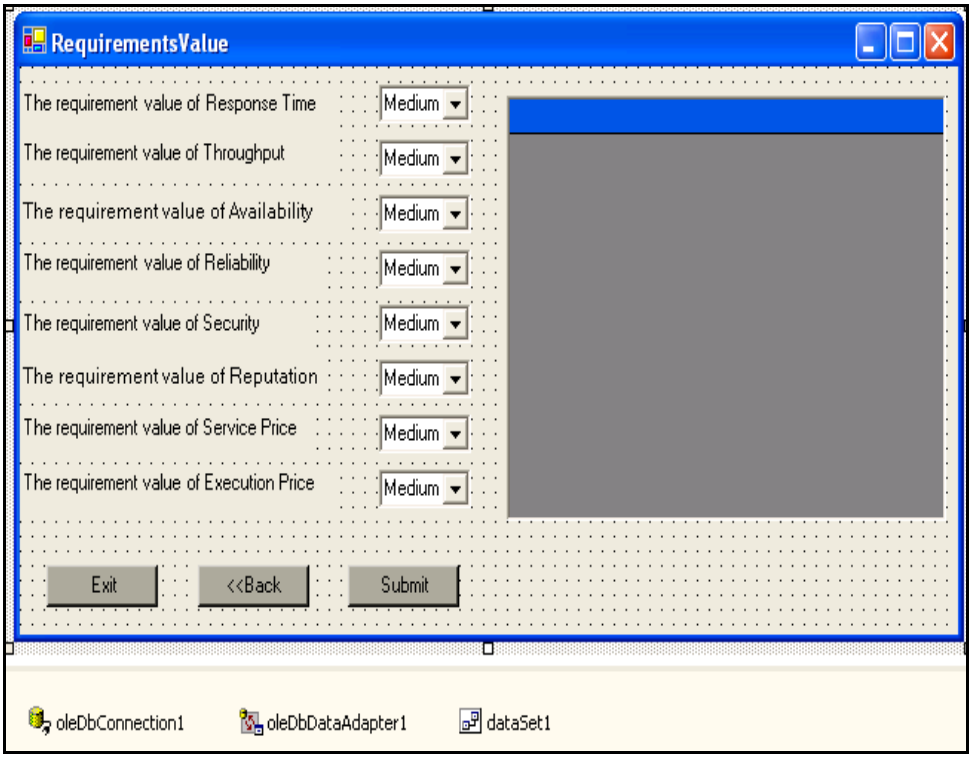

Figure B-5 RequirementsValue Form

Figure B-5

```
namespace ServiceSelection2
{
  public class RequirementsValue : System.Windows.Forms.Form
  {
  public System.Windows.Forms.Label label1;
  public System.Windows.Forms.Label label2;
  public System.Windows.Forms.Label label3;
  public System.Windows.Forms.Label label4;
  public System.Windows.Forms.ComboBox comboBox1;
  public System.Windows.Forms.ComboBox comboBox2;
  public System.Windows.Forms.ComboBox comboBox3;
  public System.Windows.Forms.ComboBox comboBox4;
  private System.Windows.Forms.Button button1;
  private System.Windows.Forms.Button button2;
  private System.Windows.Forms.Button button3;
  public System.Windows.Forms.Label label5;
  public System.Windows.Forms.ComboBox comboBox5;
  public System.Windows.Forms.Label label6;
  public System.Windows.Forms.ComboBox comboBox6;
  public System.Windows.Forms.Label label7;
  public System.Windows.Forms.ComboBox comboBox7;
  public System.Windows.Forms.Label label8;
```

```
  public System.Windows.Forms.ComboBox comboBox8;
  public System.Windows.Forms.Label label9;
  Utilities utility=new Utilities();
  ComboBox[] boxArray=new ComboBox[8];
  double[] val=new double[8];
  double values;
  public System.Windows.Forms.Label label10;
  private System.Data.OleDb.OleDbConnection oleDbConnection1;
  private System.Data.OleDb.OleDbDataAdapter oleDbDataAdapter1;
  private System.Data.OleDb.OleDbCommand oleDbSelectCommand1;
  private System.Data.OleDb.OleDbCommand oleDbInsertCommand1;
  private System.Data.DataSet dataSet1;
  private System.Windows.Forms.DataGrid dataGrid1;
  private System.Windows.Forms.ListBox listBox1;
  private System.Windows.Forms.TextBox textBox1;

  public RequirementsValue()
  {
    InitializeComponent();
    boxArray[0]=comboBox1;
    boxArray[1]=comboBox5;
    boxArray[2]=comboBox2;
    boxArray[3]=comboBox6;
    boxArray[4]=comboBox7;
    boxArray[5]=comboBox3;
    boxArray[6]=comboBox4;
    boxArray[7]=comboBox8;
     }

// Convert the sub-criteria requirements values or levels (High,
//Medium, or Low)for Response Time,Throughput, Availability,
//Reliability, Security, Reputation, Service Price and Execution
//Price from string to double values based on the sub-criteria
type and the service //domain

// convert Response Time requirement value from string to double
//type
private double responseConvert( ComboBox box)
{
  if(box.Visible==true)
  {
    string scale=(String)box.SelectedItem;
    //range of High values: High=<1msec
    if (scale.Equals( "High")==true)
      values=1;
```

```
    //Medium=[2-20]msec
    else if(scale.Equals( "Medium")==true)
      values=15;

    //Low=>=20msec
    else if(scale.Equals( "Low")==true)
      values=30;
  }
  return values;
}
// convert Throughput requirement value from string to double
//type

private double thptConvert( ComboBox box)
{
  if(box.Visible==true)
  {
    string scale=(String)box.SelectedItem;
  //range of High values: High=>20 req/sec

    if (scale.Equals( "High")==true)
      values=20;

  //Medium=[10-20]req/sec
    else if(scale.Equals( "Medium")==true)
      values=15;

  //Low<=15 req/sec
    else if(scale.Equals( "Low")==true)
      values=5;
  }
  return values;
}
// convert Availability and Reliabilityt requirement value from
string to double //type

private  double avalRelConvert( ComboBox box)
{
  if(box.Visible==true)
  {
    string scale=(String)box.SelectedItem;
  //range of High values: High [80-100]
    if (scale.Equals( "High")==true)
      values=98;

  //Medium=[50-80]
```

```
    else if(scale.Equals( "Medium")==true)
      values=70;

  //Low=<50
    else if(scale.Equals( "Low")==true)
      values=40;
  }
  return values;
}
// convert Security and Reputation requirement value from string
to double //type

private double secRepConvert( ComboBox box)
{
  if(box.Visible==true)
  {
    string scale=(String)box.SelectedItem;
  //range of High values: [4-5]
    if (scale.Equals( "High")==true)
      values=4.5;

  //Medium=[2.5-4]
    else if(scale.Equals( "Medium")==true)              values=3;

  //Low=[1-2.5]
    else if(scale.Equals( "Low")==true)
      values=2;
  }
  return values;
}
// convert Service Price requirement value from string to double
//type

private  double serPriceConvert( ComboBox box)
{
  if(box.Visible==true)
  {
    string scale=(String)box.SelectedItem;
  //range of High values: High=>60 Pound

    if (scale.Equals( "High")==true)
      values=60;

  //Medium=[30-60]Pound
    else if(scale.Equals( "Medium")==true)
      values=40;
```

```
  //Low<=30 Pound
    else if(scale.Equals( "Low")==true)
      values=15;
  }
  return values;
}
// convert Execution Price requirement value from string to double
//type

    private double execPriceConvert( ComboBox box)
    {
      if(box.Visible==true)
      {
        string scale=(String)box.SelectedItem;
      //range of High values: High=>8Pound

        if (scale.Equals( "High")==true)
          values=8;

      //Medium=[4-8]Pound
        else if(scale.Equals( "Medium")==true)
          values=5;

      //Low=[1-4]Pound
        else if(scale.Equals( "Low")==true)
          values=2;
      }
      return values;
    }

  //Convert the selected sub-criteria levels(H,M,or,L) into values and store it
  //in an array val1[]
private double[] requirementArray( )
{
  if(boxArray[0].Visible==true)
    val[0]=responseConvert(boxArray[0]);
  if(boxArray[1].Visible==true)
    val[1]=thptConvert(boxArray[1]);
  if(boxArray[2].Visible==true)
    val[2]=avalRelConvert(boxArray[2]);
  if(boxArray[3].Visible==true)
    val[3]=avalRelConvert(boxArray[3]);
  if(boxArray[4].Visible==true)
    val[4]=secRepConvert(boxArray[4]);
```

```
  if(boxArray[5].Visible==true)
    val[5]=secRepConvert(boxArray[5]);
  if(boxArray[6].Visible==true)
    val[6]=serPriceConvert(boxArray[6]);
  if(boxArray[7].Visible==true)
    val[7]=execPriceConvert(boxArray[7]);
  ArrayList list = new ArrayList();
  for(int i=0; i<val.Length;i++)
  {
    if(val[i]!=0)
    {
      list.Add(val[i]);
    }
  }
  double[] val1=(double[]) list.ToArray(typeof (double));
  return val1;
}

// Retreive the data result by sending a query request based on the service
//requester's sub-quality level (H,M, or L)to the Amazon database and matching

//between the requirement values or levels and the provided quality specifications
//which stored in the Amazon database
private void dataRetreive()
{
// For simplicity, the sub-criteria types selected are:Availability, Reputaion and/or
Service Price
  if(boxArray[2].Visible==true&& boxArray[5].Visible==true &&
        boxArray[6].Visible==true)
  {
// if Availability's level is High, Reputaion's level is High and Service Price's level
is High
  if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("High")&&boxArray[6].SelectedItem.Equals("High"))
    {
//Quality Matchmaker sends a query to Amazon database by matching
//between the quality requirement values or levels specified by the service

//requester and the quality specification provided by the service provider

        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
```

```
        AND 100 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price>=60";
    }
// if Availability's level is Medium, Reputaion's level is Medium and Service
Price's level is Medium
        else if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Medium")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 OR AmazonTable.SellerReputation BETWEEN 2.5 AND 4 AND
        AmazonTable.Price BETWEEN 30 AND 60";
       }
// if Availability's level is Low, Reputaion's level is Low and Service Price's level
is Low
        else if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Low")&&boxArray[6].SelectedItem.Equals("Low"))

    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 OR
        AmazonTable.SellerReputation BETWEEN 1 AND 2.5 AND
        AmazonTable.Price<=30";
       }
// if Availability's level is High, Reputaion's level is High and Service Price's level
is Medium
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("High")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is High, Reputaion's level is High and Service Price's level
is Low
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("High")&&boxArray[6].SelectedItem.Equals("Low"))
```

```
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price<=30";
    }
// if Availability's level is High, Reputaion's level is Medium and Service Price's
level is High
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Medium")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 4
        AND AmazonTable.Price>=60";
    }
// if Availability's level is High, Reputaion's level is Medium and Service Price's
level is Medium
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Medium")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 4
        AND AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is High, Reputaion's level is Medium and Service Price's
level is Low
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Medium")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 4
        AND AmazonTable.Price<=30";
    }
// if Availability's level is High, Reputaion's level is Low and Service Price's level
is High
```

```
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Low")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price>=60";
    }
// if Availability's level is High, Reputaion's level is Low and Service Price's level
is Medium
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Low")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is High, Reputaion's level is Low and Service Price's level
is Low
        if
        (boxArray[2].SelectedItem.Equals("High")&&boxArray[5].SelectedItem.
        Equals("Low")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 80
        AND 100 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price<=30";
    }
// if Availability's level is Medium, Reputaion's level is High and Service Price's
level is High
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("High")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price>=60";
```

```
    }
// if Availability's level is Medium, Reputaion's level is High and Service Price's
level is Medium
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("High")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is Medium, Reputaion's level is High and Service Price's
level is Low
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("High")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 4 AND 5
        AND AmazonTable.Price<=30";
    }
// if Availability's level is Medium, Reputaion's level is Medium and Service
Price's level is High
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Medium")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 4
        AND AmazonTable.Price>=60";
    }
// if Availability's level is Medium, Reputaion's level is Medium and Service
Price's level is Low
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Medium")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
```

```
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 2.5 AND 4
        AND AmazonTable.Price<=30";
    }
// if Availability's level is Medium, Reputaion's level is Low and Service Price's
level is High
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Low")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price>=60";
    }
// if Availability's level is Medium, Reputaion's level is Low and Service Price's
level is Medium
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Low")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is Medium, Reputaion's level is Low and Service Price's
level is Low
        if
        (boxArray[2].SelectedItem.Equals("Medium")&&boxArray[5].SelectedIte
        m.Equals("Low")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability BETWEEN 50
        AND 80 AND AmazonTable.SellerReputation BETWEEN 1 AND 2.5
        AND AmazonTable.Price<=30";
    }
// if Availability's level is Low, Reputaion's level is High and Service Price's level
is High
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("High")&&boxArray[6].SelectedItem.Equals("High"))
```

```
            {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 4 AND 5 AND
        AmazonTable.Price>=60";
    }
// if Availability's level is Low, Reputaion's level is High and Service Price's level
is Medium
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("High")&&boxArray[6].SelectedItem.Equals("Medium"))
       {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 4 AND 5 AND
        AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is Low, Reputaion's level is High and Service Price's level
is Low
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("High")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 4 AND 5 AND
        AmazonTable.Price<=30";
    }
// if Availability's level is Low, Reputaion's level is Medium and Service Price's
level is High
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Medium")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 2.5 AND 4 AND
        AmazonTable.Price>=60";
    }
// if Availability's level is Low, Reputaion's level is Medium and Service Price's
level is Medium
```

```
                if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Medium")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 2.5 AND 4 AND
        AmazonTable.Price BETWEEN 30 AND 60";
    }
// if Availability's level is Low, Reputaion's level is Medium and Service Price's
level is Low
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Medium")&&boxArray[6].SelectedItem.Equals("Low"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 2.5 AND 4 AND
        AmazonTable.Price<=30";
    }
// if Availability's level is Low, Reputaion's level is Low and Service Price's level
is High
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Low")&&boxArray[6].SelectedItem.Equals("High"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle, ProductAvailability, SellerReputation, Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 1 AND 2.5 AND
        AmazonTable.Price>=60";
    }
// if Availability's level is Low, Reputaion's level is Low and Service Price's level
is Medium
        if
        (boxArray[2].SelectedItem.Equals("Low")&&boxArray[5].SelectedItem.E
        quals("Low")&&boxArray[6].SelectedItem.Equals("Medium"))
    {
        oleDbDataAdapter1.SelectCommand.CommandText="SELECT SellerID,
        ProductTitle,ProductAvailability,SellerReputation,Price FROM
        AmazonTable WHERE AmazonTable.ProductAvailability<=50 AND
        AmazonTable.SellerReputation BETWEEN 1 AND 2.5 AND
        AmazonTable.Price BETWEEN 30 AND 60";
```

```
    }
  }
  dataSet1.Clear();
//Fill dataSet1 with result from the query

  oleDbDataAdapter1.Fill(dataSet1, "AmazonTable");

//create a datatable named dataTable and assign it the collection of data stored
//by dataSet1. The Tables columns contains: SellerID, ProductTitle,
//ProductAvailability, Seller Reputaion,and Price fields
  DataTable dataTable=dataSet1.Tables[0];

  Matrix criteriaOffered=new Matrix(dataSet1.Tables[0].Columns.Count-

          2,dataSet1.Tables[0].Rows.Count);
  string [,] col=new string [
          dataSet1.Tables[0].Rows.Count,dataSet1.Tables[0].Columns.Count];
// if no result retreived then the service requester has to specify new requirements
//of quality levels
  if(dataSet1.Tables[0].Rows.Count==0)
  {
    MessageBox.Show("There are no results relating to your criteria, please try
again", "message Box", MessageBoxButtons.OK);
  }
// if result retrieved from the matching between the quality requirement and
quality specification
   else
  {
    for(int j=0; j<dataSet1.Tables[0].Columns.Count;j++)
    {
      for(int i=0; i<dataSet1.Tables[0].Rows.Count; i++)
      {
      col[i,j]=(dataTable.Rows[i][j].ToString());
      listBox1.Items.Add(col[i,j]);
      }
    }
//store the result from the dataset and put it in the Performance matrix

// called criteriaOffrered[,], which will be used for Euclidean distance calculation

  for(int i=0; i<dataSet1.Tables[0].Columns.Count-2;i++)
    {
      for(int j=0; j<dataSet1.Tables[0].Rows.Count;j++)
      {
```

```
//criteriaOffered [,] rows contain the sub-criteria fields and the columns contain
the service records
      criteriaOffered[i,j]=(Double.Parse(col[j,i+2]));
      textBox1.AppendText("the matrix["+i+","+j+"] is
                          "+criteriaOffered[i,j]+"  "+"\n");
      }
    }
  double []reqArray=requirementArray();

// calculate the Euclidean distance for each service by calling EuclideanDistance

// method from Utilities class
          double[]EuclDistance=utility.EuclideanDistance(criteriaOffered,SubCriter
          iaSelection.totalSubNum,dataSet1.Tables[0].Rows.Count,SubCriteriaSelec
          tion.totalWeight,reqArray);

// Rank the result services based on Euclidean distance value
// the rank is from the smallest distance to the largest one
  for(int i=1; i<=EuclDistance.Length-1; i++)
  {
    if(EuclDistance[i-1]>EuclDistance[i])
    {
      double temp = EuclDistance[i-1];
      EuclDistance[i-1]=EuclDistance[i];
      EuclDistance[i]=temp;
    }
  }
  DataTable table;
  DataColumn serviceProvider = new DataColumn("Service provider");
  DataColumn serviceName = new DataColumn("Service name");
  DataColumn distance = new DataColumn("Service distance");
//create table called Services which contains:Service provider, Service name,

//and Service distance fields for each service's record

  table = new DataTable("Services");
  table.Columns.Add(serviceProvider);
  table.Columns.Add(serviceName);
  table.Columns.Add(distance);
  for(int j=0; j<dataSet1.Tables[0].Rows.Count; j++)
  {
  textBox1.AppendText("the matching distance of "+ col[j,0]+"   " + col[j,1] +"
"+  "is: "+Math.Round(EuclDistance[j],3)+" \n\n");
    DataRow row;
    row=table.NewRow();
```

```
    row["service provider"]= col[j,0];
    row["Service name"]= col[j,1];
    row["Distance"]= Math.Round(EuclDistance[j],3);
    table.Rows.Add(row);
   }
// create a new DataSet object named dataset2
   DataSet dataset2 = new DataSet();

//add the new table to the dataset's Tables

   dataset2.Tables.Add(table);

// bind the new dataset to a data grid to display the final result

   dataGrid1.SetDataBinding(dataset2, "Services");
   }
}
   private void Submit_Click(object sender, System.EventArgs e)
   {
// call the dataRetreive method which display the ranked services' result
// based on Euclidean distance
   dataRetreive();
   }

   private void Back_Click(object sender, System.EventArgs e)
   {
     this.Close();
   }
   private void Exit_Click(object sender, System.EventArgs e)
   {
     base.Dispose();
   }
}
}
```

Appendix C: ADO.NET and Access Database

This Appendix describes the connection to an Access database called Amazon, using *oleDbConnection1*. The query result data is retrieved and displayed in the DataGrid using *oleDbDataAdapter1* and *dataSet1* with ADO.NET (ActiveX Data Objects.NET.

The following steps describe how to connect to an Access database from Windows application:

Step-1: Create a Data Connection:

- Server Explorer establish the connections to databases by right-click Data Connections and choose Add Connection as shown in the screenshot below:

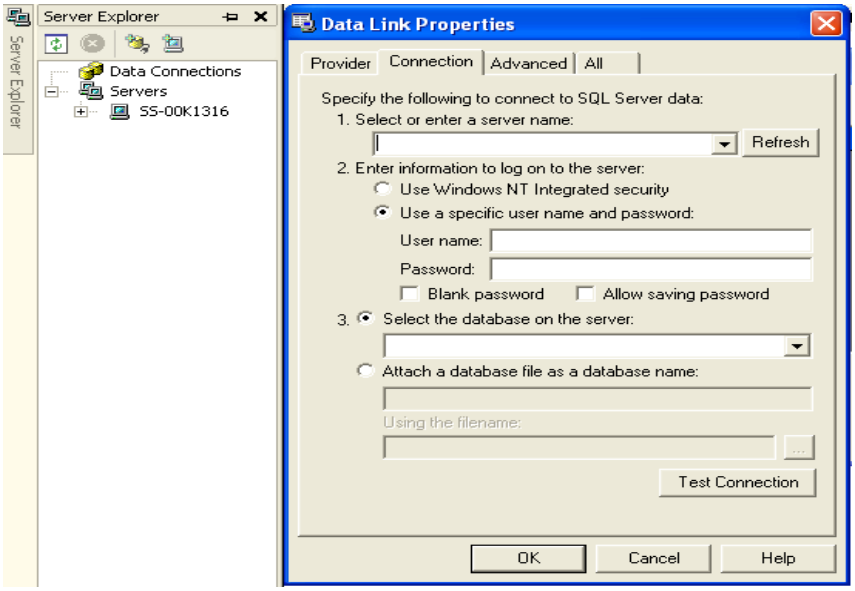

Figure C- 1Screenshot of Data Connections

- In the Provider tab Microsoft Jet 4.0 OLE DB Provider is selected for connecting to an Access database and then clock Next, as shown in the screenshot below.

Appendix C: ADO.NET and Access Database

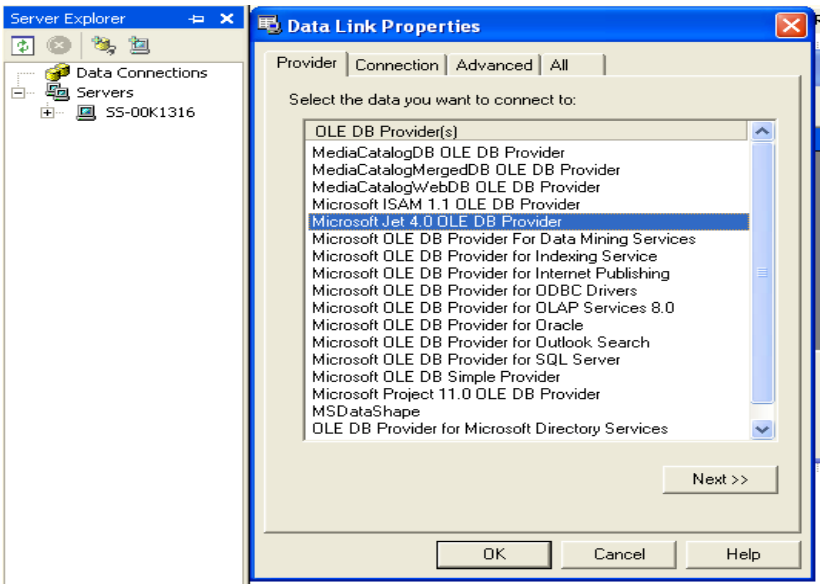

Figure C- 2 Screenshot of Selecting Database Provider

- Enter the Access database path and click 'Test Connection' button. It appears 'Test connection succeeded' as shown in the screenshot below.

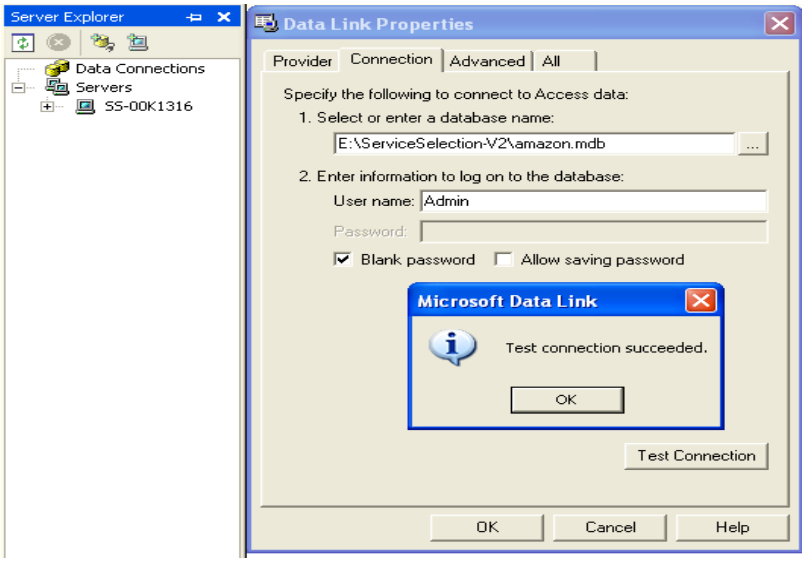

Figure C- 3 Screenshot of Access Database Connection

- Then the screenshot below browse the Access file database which called "Amazon.mdb.Admin".

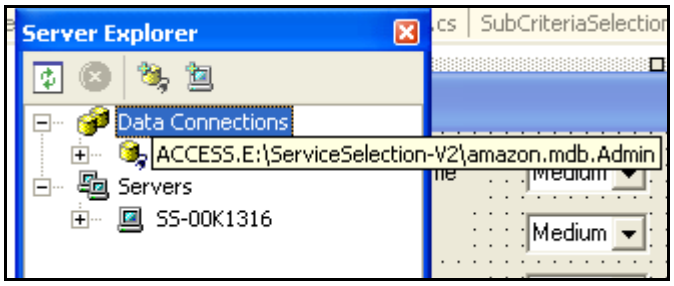

Figure C- 4 Access File Database

- Drag the Access file path "Access.E\erviceSelection-V2\amazon.mdb.Admin" and drop it on the *RequirementsValue* form and *oleDbConnection1* will appear below the form as shown in Figure C- 9.

Step-2 Create a Data Adapters

After creating a connection to the database, it needs to create a data adapter with appropriate SQL statement for managing the connection and retrieving the result of query from the data source [148].

- From the "Data" group "OleDbDataAdapter" is dragged and the "Data Adapter Configuration Wizard" starts as shown in the two screenshots below.

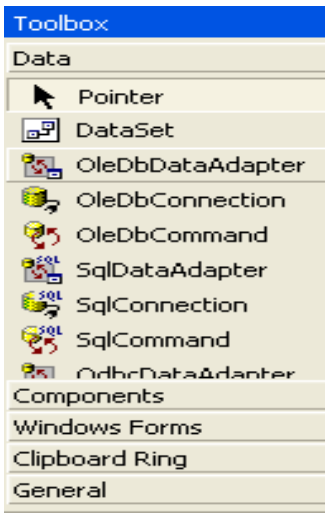

Figure C- 5 Screenshot of dragging OleDbDataAdapter from Data group

Figure C- 6 Screenshot of Data Adapter Configuration Wizard

- Data adapter uses SQL statements to access the Access database as shown in the screenshot below.

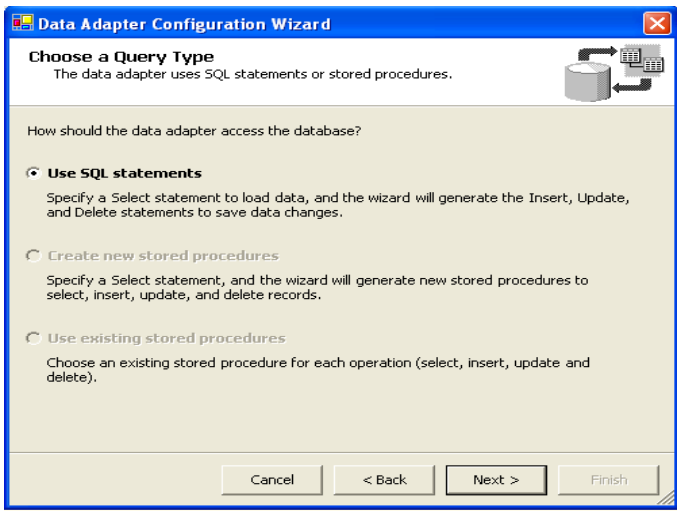

Figure C- 7 Screenshot of Choosing Query Type

- Create SQL statement that selects all the columns in the AmazonTable table as in the following:

SELECT AmazonTable.*

FROM AmazonTable

As shown in the screenshot below

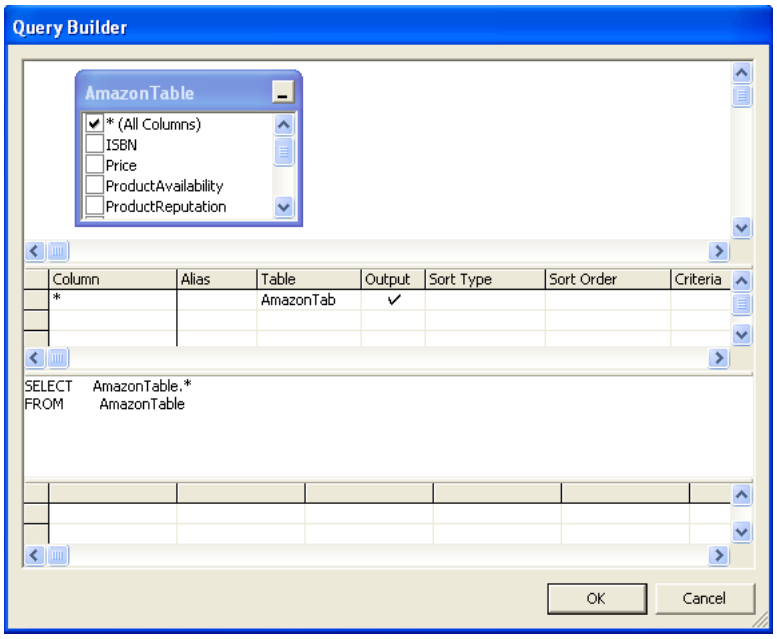

Figure C- 8 Screenshot of Generating SQL Statement

- Then *oleDbDataAdapter1* will appear below the *RequirementsValue* form as shown in Figure C- 8.

Step-3 Create Dataset

DataSets are used to store the query results and display the result using DataGrid

- Drag the DataSet from Data group and drop it in the *RequirementsValue* form then *dataSet1* object will appear below the form as shown in Figure C- 9.

Appendix C: ADO.NET and Access Database

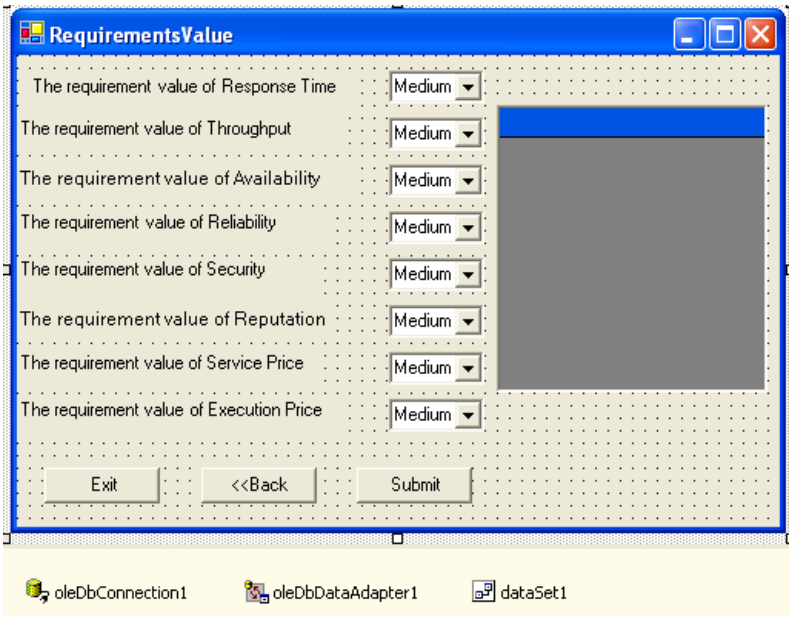

Figure C- 9 Screenshot shows oleDbConnection1, oleDbDataAdapter1 and dataSet1 objects

Appendix D: Amazon Web Services (AWS) Case Study

D-1 What is Amazon Web Services (AWS)?

Web services can be defined as the Web-based applications that dynamically interact with other Web applications using XML-based open standards such as SOAP, UDDI and WSDL. Web services are a way of accessing information or services over the Web. The requester makes a specific request to a server for a type of information, and the server returns the information in some form. Microsoft's .NET and Sun's Sun ONE (J2EE) are the major development platforms that support these standards.

Amazon.com was debuted Amazon Web Services (AWS) in July 2002, announcing that the service can use XML-based Web services technology to make the contents of its catalog (database) freely available for use by any Web site or software application.

Amazon Web Services (AWS) [134] is Amazon API (Application Program Interface). An API is a set of building blocks made up of routines, protocols, and tools that influence how users interface with the service. AWS offers applications that range from retrieving information about a set of products, vendors, and transactions to adding a product to a shopping car, wish list, or registry. Figure D-1 illustrates the interactions between Amazon Web Services (AWS) and its customers The Amazon's customer (such as buyers, sellers (merchants who sell on Amazon's platform), Web site owners (associates), and developers (people who use Amazon's Web services)) can access the Amazon Web Services using either XML over HTTP (REST) or a remote procedure call API with a Simple Object Access Protocol (SOAP) interface. Both of these methods return structured data (product name, manufacturer, price, etc.). Only about 15% of Amazon Web

Services calls are made with SOAP and the remainder with REST. Amazon.com has provided a Web Services Definition Language (WSDL) file, which contains the definition of the Web service. A developer with access to this WSDL file can write a client application to use the Amazon Web Services.

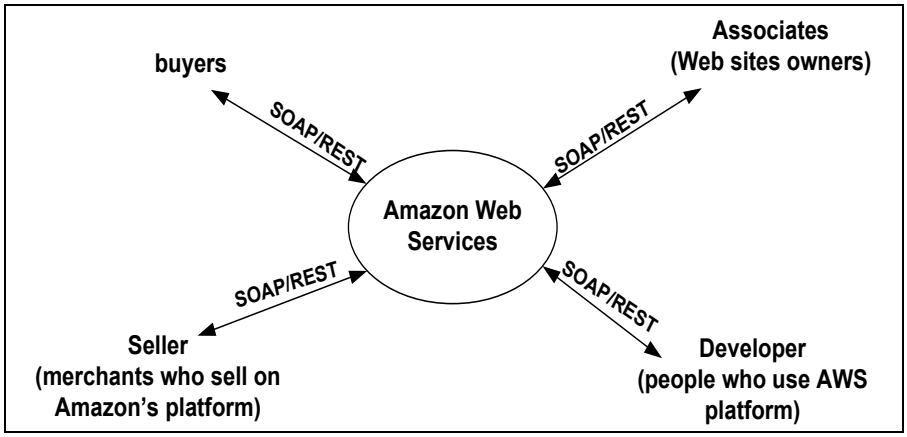

Figure D- 1 Relationship between Amazon Web Services (AWS) and its Customers

D-2 Benefits of Using Amazon Web Services

Here are some of the key benefits to using Amazon Web Services:

- Scalable product integration: AWS enables the customers to add much of the rich product content that makes Amazon.com a great place to shop, such as real-time pricing and availability, product images, customer reviews, product descriptions, sales rank, and more. This content enable Associates to create and display full product detail pages that provide visitors the information they need to make a purchasing decision

- Flexible merchandising: Product content can be integrated into the look and design of client Web site.

- Product Search: Enable the visitors to conduct product searches across all major product categories available at Amazon.com. Product search results can be embedded directly into client Web site.

- Remote Shopping Cart: Enable the visitors to add products into the Amazon shopping cart while on the associates' site.

- AWS is free to join and use.

So Amazon's aims in providing Web services were, to support industry standards, provide remote access to data and functionality, and to create a software development platform (AWS platform) to create websites and applications that perform various functions, such as enabling and completing transactions, retrieving information about Amazon products or adding a product to an Amazon shopping cart, wish list, or registry.

Amazon Web Services (AWS) provides E-Commerce Service (ECS) Version 3.0/4.0. Amazon E-Commerce Service is explained in the following section.

D-3 Amazon E-Commerce Service (ECS)

Amazon E-Commerce Service (ECS) [149] exposes Amazon's product data and E-Commerce functionality which allowing developers, Web site owners and merchants to leverage the data and functionality that Amazon uses to power its own E-Commerce business. ECS 4.0, which has launched on October 4th, 2004, is available free-of-charge, makes it extremely easy for developers to build rich Web sites and applications.

D-3-1 E-Commerce Service (ECS) Features

With ECS 4.0, developers can add rich content and powerful capabilities to Web sites and applications by using the following features:

Detailed Product Information on all Amazon.com Products

ECS 4.0 provides detailed product and pricing information for all products across every product category in the Amazon.com product catalog. It provides access to product attributes, which used to describe and differentiate products on a Web site

or in an application. For example, developers can obtain the color, luster, size, and clarity of a pearl sold in Amazon.com's Jewelry store.

Access to Amazon.com Product Images

ECS 4.0 provides access to images in all product categories, even for the newest product categories Amazon.com has launched.

All Customer Reviews associated with a Product

ECS 4.0 allows developers to retrieve all customer reviews for a specific product, which used to enhance the richness of Web sites and applications.

Extended Search

With ECS 4.0, developers can create applications with more complex search options than before. Previously, developers were limited to a simple keyword search. Now developers can build on the same functionality as Amazon.com's "Advanced Search," which allows searching by numerous attributes, including brand, price, and category.

Remote Shopping Cart

ECS 4.0 allows developers to add Remote Shopping Cart functionality to their own Web site or application. With this Remote Shopping Cart functionality, developers can add items to an Amazon.com shopping cart and submit it to Amazon.com for check-out processing.

Amazon Wish List Search

Developers can now add wish list search by name, email address, city, and state into their applications.

Precise Response Groups

ECS 4.0 introduces Response Groups; a new feature that allows developers to specify and retrieve only the information they want from Amazon.com. This approach is far more flexible and efficient than the previous response types (lite (some data) or heavy (more data). ECS 4.0 includes more than 30 response groups that developers can mix and match to get exactly the information they want.

Multi-Operation and Batch Interfaces

ECS 4.0 enables developers to input a single request and receive responses that include data from up to two operations, which means more data from fewer requests and faster application performance.

Amazon E-Commerce (ECS) provides ECS 4.0 SDK documentation guide which provides all the information the developer needs to create Web sites or applications that integrate ECS as well as diagnose and resolve the problems**.** ECS 4.0 SDK documentation guide [48] is described in the following section.

D-4 Amazon E-Commerce Service (ECS) 4.0 Software Development Kit (SDK)

ECS 4.0 SDK provides all the information the developer needs to create Web sites or applications that integrate ECS as well as diagnose and resolve the problems.

In order to access ECS, the developer must first register with the Amazon Web Services program. Registration is free. The developer will be assigned a subscription ID after completing the registration that will allow him to access all ECS functionality.

D-4-1 Introduction to Amazon E-Commerce Service (ECS)

ECS is an Application Programming Interface (API) that allows the requester to access Amazon data and functionality through a Web site or Web-enabled application. ECS follows the standard Web services model: the requester requests

data through REST (XML over HTTP) or SOAP and data is returned by the service as an XML-formatted stream of text.

ECS is currently incorporated in thousands of Web sites and applications around the world. Amazon partners use ECS for competitive pricing, inventory management, and other online retailing tasks.

ECS is available for all Amazon sites (or locales):

- US (amazon.com)
- UK (amazon.co.uk)
- Germany (amazon.de)
- Japan (amazon.co.jp)
- France (amazon.fr)
- Canada (amazon.ca)

D-4-2 Selecting a Web Services Access Method

There are two options for accessing ECS:

- Making REST requests
- Making SOAP requests

Making REST Requests

This section explains how to use REST (Representational State Transfer) to make requests through Amazon E-Commerce Service (ECS). REST is a Web services protocol that was created by Roy Fielding in his Ph.D. thesis ([48] cited [150]).

REST allows the user to make calls to ECS by passing parameter keys and values in a URL (Uniform Resource Locator). ECS returns its response in XML (Extensible Markup Language) format. The developer can enter the REST URL into the browser's address bar, and the browser displays the raw XML response.

Request Parameters

Appendix D: Amazon Web Services (AWS) Case Study

The REST request to ECS begins with a base URL which is specific to the locale in which the requester wants to make the request. The following base URLs are available:

For Amazon.com (US)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService

For Amazon.co.uk (UK)

http://webservices.amazon.co.uk/onca/xml?Service=AWSECommerceService

The base URL is followed by a series of request parameters. Parameters are separated from each other by an ampersand (&) character. Each parameter consists of a key and a value, separated from each other by an equals sign (=). The parameters and their values are case-sensitive; for example, Operation=ItemSearch works correctly, but operation=itemsearch produces an error.

The following example shows a simple REST request that searches for books on Amazon.com.

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=[your subscription ID here]
&Operation=ItemSearch
&SearchIndex=Books
&Keywords=dog
The parameters in the example are described below:

SubscriptionId=[your subscription ID here]

SubscriptionId is required in all ECS requests. After registering as an Amazon Web Services Developer, the requester will be assigned a subscription ID which allows him to access all ECS functionality.

Operation=ItemSearch

Operation is required in all ECS requests; it tells ECS what action it must perform. In the example, the operation is ItemSearch, which tells ECS to perform a search for products in the Amazon.com catalog that meet particular criteria.

SearchIndex=Books

SearchIndex is required by the ItemSearch operation. SearchIndex tells the ItemSearch operation what type of product to search for. The example searches through the Books index.

Keywords=dog

Keywords tells the ItemSearch operation to search the Amazon.com catalog for specific text values. In the example, the request searches for the word "dog."

The requester can search for more than one keyword separated by URL-encoded space characters (%20). For example, to search for cats and dogs, the requester specifies *Keywords=cats%20dogs* in the request.

Controlling Return Data with Response Groups

The requester/developer can control the amount and what kinds of data are returned in a response by specifying the *ResponseGroup* parameter. If he does not specify the *ResponseGroup* parameter, ECS returns a default response groups (*Request* and *Small* response groups as in the previous example), depending on the operation he uses. He can specify more than one response group, separated by commas, in order to refine and tailor response data to fit the needs of his application.

The *Request* response group returns the list of parameters and values that he has requested. *Request* is a default response group for every operation.

The *Small* response group returns global, item-level data about items included in the response. For example, the item's Amazon Standard Item Number (ASIN), name, creator (for example, author or artist), product group, URL, and manufacturer. The requester/developer can expand the information returned by

specifying *Medium* or *Large* response group. He can also narrow the response to include specific information about each item by specifying response groups like *Images* or *Accessories*. The *Response Groups* will be described in details in the coming sections.

The following example uses the *ItemIds* response group to retrieve only the ASINs for books about dogs:

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[your subscription ID here]*
&Operation=ItemSearch
&SearchIndex=Books
&Keywords=dog
&ResponseGroup=ItemIds

Making SOAP Requests

SOAP (Simple Object Access Protocol) allows third-party developers to use Amazon E-Commerce Service (ECS) by making remote procedure calls. This information is encoded using XML (Extensible Markup Language). ECS publishes a Web Services Description Language (WSDL) document that defines all the available ECS APIs, their parameters, and the data that they return.

The SOAP End Points

For Amazon.com (US) data

http://webservices.amazon.com/onca/soap?Service=AWSECommerceService

For Amazon.co.uk (UK) data

http://webservices.amazon.co.uk/onca/soap?Service=AWSECommerceService

D-4-3Amazon E-Commerce (ECS) Operations

Amazon E-Commerce Service (ECS) operations allow the requester/developer to access the information available on Amazon's Web site.

ECS operations fall into two categories. Search operations, whose names end in "Search," allow the requester/developer to query an Amazon Web site for content or data using keywords, titles, creator names, or other information. Lookup operations, whose names end in "Lookup," allow the requester/developer to request content or data keyed by an ID such as an ASIN (Amazon Standard Item Number), a UPC (Universal Product Code), a wish list ID, or a seller ID.

All Operations

The following operations are available in Amazon E-Commerce Service:

- BrowseNodeLookup
- CartAdd
- CartClear
- CartCreate
- CartGet
- CartModify
- CustomerContentLookup
- CustomerContentSearch
- Help
- ItemLookup
- ItemSearch
- ListLookup
- ListSearch
- SellerListingLookup
- SellerListingSearch
- SellerLookup

- SimilarityLookup
- TransactionLookup

ItemLookup and *ItemSearch* operations will be explained in the following. Further information about the remaining operations can be found in [48].

ItemLookup Operation

Description

The ItemLookup operation allows the requester/developer to retrieve catalog information for up to ten products or restaurants (US only). ItemLookup provides access to customer reviews, variations, product similarities, pricing, availability, images, product accessories, and other information.

Sample Request

Using ItemLookup (REST)

The following *ItemLookup* example demonstrates a request for item information for an ASIN.

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[ID]*
&Operation=ItemLookup
&ItemId=*[An ASIN]*

Request Parameters

Request parameters specify the terms of the requester/developer request and control the output data that is returned to him.The *required* parameters must be include in every request.

The following parameters are specific to the *ItemLookup* operation:

Table D- 1ItemLookup Request Parameters

Parameter	Description	Required?	Value
Operation	The operation	Always Required	**ItemLookup**
ItemId	Product(s) the requester/developer would like information about. By default the item IDs are assumed to be ASINs, unless he	Always Required	**Product IDs**

	specifies the IdType parameter.		
IdType	Type of product ID the requester/developer is requesting information about. SKU requires a MerchantId. US only. UPC is US only. EAN is the same as JAN (Japanese article Number) If the requester/developer selects SKU, UPC, or EAN as the IdType for his request, he also needs to include the *SearchIndex* parameter.	Required for SKU, UPC or EAN search	Default Value • **ASIN** Valid Values • **ASIN** • **SKU (US only)** • **UPC (US only)** • **EAN (DE/JP only)**
SearchIndex	The Amazon store to search. This parameter is ignored for ASIN searches. *SearchIndex* is required any time the requester/developer selects SKU, UPC, or EAN as the IdType for his request	Required for UPC, SKU or EAN searches	Valid Values: • **Electronics** •**Music** • **Classical** • **DVD** • **VHS** • **Video** • **OutdoorLiving** •**HealthPersonal-Care** • **Kitchen** • **Software** • **SoftwareVideo-Games** • **VideoGames** • **Tools**
Condition	Use the *Condition* parameter to filter the offers returned in the product list by condition type.	Always Optional	Default Value • **New** Valid Values • **All** • **New** • **Used** • **Refurbished** • **Collectible**
DeliveryMethod	Use the *DeliveryMethod* parameter to filter offers returned in the product list by delivery method. Valid values are Ship and ISPU (In-store pickup).	Always Optional	Default Value • **Ship** Valid Values • **Ship** • **ISPU**
ResponseGroup	Controls the data returned by the operation. Use this parameter to specify which response group(s), or group(s) of data elements that wiuld be returned. The requester/developer can specify as many response groups as he wishes using a comma-separated list (REST) or multiple elements (SOAP).	Always Optional	Default Values • *Request* • *Small* Valid Values • *Request* • *ItemIds* • *Small* • *Medium* • *Large*

			• *OfferFull* • *Offers* • *OfferSummary* • *Variations* •*VariationMinimum* •*VariationSummary* • *ItemAttributes* • *Tracks* • *Accessories* • *EditorialReview* • *SalesRank* • *BrowseNodes* • *Images* • *Similarities* • *Reviews* • *ListmaniaLists*

ItemSearch Operation

Description

The ItemSearch operation allows the requester/developer to search for products and restaurants.

Sample Request

Using ItemSearch (REST)

The following ItemSearch example demonstrates a keyword search within a specified index. It also returns the search results in the order specified by the sort that is entered.

http://webservices.amazon.com/onca/
xml?Service=AWSECommerceService&SubscriptionId=*[Your Subscription ID Here]*&Operation=ItemSearch&Keywords=*[A Keywords String]*&SearchIndex=*[A Search Index String]*&Sort=*[A Sort String]*

Request Parameters

Request parameters specify the terms of our request and control the output data that is returned to the requester/developer.

The following parameters are specific to the ItemSearch operation:

Table D- 2 itemSearch Request Parameters

Parameter	Description	Required?	Value
Operation	The operation	Always	**ItemSearch**

		Required	
SearchIndex	The list of available *SearchIndex* values, listed by locale.	Always Required	Valid Values: • A Search Index (varies by locale)
Keywords	Amazone E-Commerce Service (ECS) will match the word or phrase that include in the request against various product fields, including product title, author, artist, description, manufacturer, etc.		Valid Value: • A KeywordsString
Title	Use the *Title* parameter when the requester/developer wants to query against product titles only.		Valid Value: • A Title String
ItemPage	This parameter returns the specified page. When we use *ItemPage*, Item-Search will return 10 search results at a time. The maximum *ItemPage* number that can be returned is 3200. If we do not include *ItemPage* in our request, the first page (containing the first 10 items or all of the items if there are less than 10) will be returned by default.	Always Optional	Default Value: • **1** Valid Values: • Integers 1 to 3200
Sort	Use the *Sort* parameter to specify how the item search results will be ordered.	Always Optional	Valid Values: •Varies by SearchIndex and Locale
MinimumPrice	Use the *MinimumPrice* parameter to set a lower price bound on products returned by ItemSearch. The *MinimumPrice* value must be specified in pennies (or equivalent in local currency).	Always Optional	Valid Value: • An Integer
MaximumPrice	Use the *MaximumPrice* parameter to set an upper price bound on products returned by ItemSearch. The *MaximumPrice* value must be specified in pennies (or equivalent in local currency).	Always Optional	Valid Value: • An Integer
Condition	Use the *Condition* parameter to filter the offers returned in the product list by condition type.	Always Optional	Default Value • **New** Valid Values • **All** • **New** • **Used** • **Refurbished** • **Collectible**
DeliveryMethod	Use the *DeliveryMethod* parameter to filter offers returned in the product list by delivery method. Valid values are Ship and ISPU (In-store pickup).	Always Optional	Default Value • **Ship** Valid Values • **Ship** • **ISPU**
ResponseGroup	Controls the data returned by the operation. Use this parameter to specify which response group(s), or group(s) of data elements that	Always Optional	Default Values • *Request* • *Small*

	would be returned. The requester/developer can specify as many response groups as he wishes using a comma-separated list (REST) or multiple elements (SOAP).		Valid Values • *Request* • *ItemIds* • *Small* • *Medium* • *Large* • *Offers* • *OfferSummary* • *Variations* • *VariationMinimum* • *VariationSummary* • *ItemAttributes* • *Tracks* • *Accessories* • *EditorialReview* • *SalesRank* • *BrowseNodes* •*Images* • *Similarities* • *Reviews* • ListmaniaLists

D-4-4 Response Groups

Response groups are data sets that can be returned by Amazon E-Commerce Service (ECS). They allow the requesters/developers to tailor their requests to return only the data they need. Each operation, such as ItemSearch or SimilarityLookup, has a list of valid response groups that can be used with it. The list of valid response groups supported for an operation is found in that operation's ResponseGroup request parameter as shown in Table D- 1 and Table D- 2.

All Response Groups

- Accessories
- BrowseNodeInfo
- BrowseNodes
- Cart
- CartSimilarities
- CustomerFull
- CustomerInfo

- CustomerLists
- CustomerReviews
- EditorialReview
- Help
- Images
- ItemAttributes
- ItemIds
- Large
- ListFull
- ListInfo
- ListItems
- ListmaniaLists
- ListMinimum
- Medium
- OfferFull
- Offers
- OfferSummary
- Request
- Reviews
- SalesRank
- Seller
- SellerListing
- Similarities
- Small

- Tracks
- TransactionDetails
- VariationMinimum
- Variations
- VariationSummary

Some of the response groups are explained and further information about the remaining is found in [48].

CustomerReviews Response Group

Description

The *CustomerReviews* response group provides the Reviews for each customer listed in the response. Each review in the response is described by the elements for the ASIN reviewed, the product rating, the review Summary, the review Comment, and DateOfReview. The product rating is used to calculate the product *Reputation* by using the equation:

$q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the customer's product rating, n is the number of times the product has been graded (see Section **Error! Reference source not found.**).

REST Sample Response and Request

Sample Response (REST)

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomerContentLookupResponse
xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>0ZX5BT4M4DEMTWK6YC76</RequestId>
<Arguments>
<Argument Name="Service" Value="AWSECommerceService"/>
<Argument Name="AssociateTag" Value="[Your Associate ID Here]"/>
<Argument Name="CustomerId" Value="A2KEKKJ9CAC2KC"/>
```

<Argument Name="SubscriptionId" Value="*[Your Subscription ID Here]*"/>
<Argument Name="ResponseGroup" Value="CustomerReviews"/>
<Argument Name="Operation" Value="CustomerContentLookup"/>
</Arguments>
</OperationRequest>
<Customers>
<Request>
<IsValid>True</IsValid>
</Request>
<Customer>
<CustomerReviews>
<Review>
<ASIN>B0000VUP40</ASIN>
<Rating>5</Rating>
<Date>1068860248</Date>
<Summary>Ridiculously Good Cookies</Summary>
</Review>

The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=CustomerContentLookup
&CustomerId=A2KEKKJ9CAC2KC
&ResponseGroup=CustomerReviews

ItemAttributes Response Group

Description

The *ItemAttributes* response group provides information about each item in the response that is unique to the item's product category (Books, DVD, Electronics, Apparel, etc.). It provides ListPrice element that include the product price which is equivalent to *Service Price* in the proposed quality criteria classification in 3.3.

REST Sample Response and Request

Sample Response (REST)

<?xml version="1.0" encoding="UTF-8"?>
<ItemLookupResponse
xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>0NSJAG4Y97K07K4SFJX0</RequestId>
<Arguments>
<Argument Name="Service" Value="AWSECommerceService"/>

<Argument Name="AssociateTag" Value="*[Your Associate ID Here]*"/>
<Argument Name="SubscriptionId" Value="*[Your Subscription ID Here]*"/>
<Argument Name="ItemId" Value="B00008OE6I"/>
<Argument Name="ResponseGroup" Value="ItemAttributes"/>
<Argument Name="Operation" Value="ItemLookup"/>
</Arguments>
</OperationRequest>
<Items>
<Request>
<IsValid>True</IsValid>
</Request>
<Item>
<ASIN>B00008OE6I</ASIN>
<ItemAttributes>
<Height Units="inches">2.24</Height>
<Length Units="inches">1.09</Length>
<ListPrice>
<Amount>44999</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$449.99</FormattedPrice>
</ListPrice>
<Manufacturer>Canon Cameras US</Manufacturer>
<NumberOfItems>1</NumberOfItems>
<ProductGroup>Photography</ProductGroup>
<Title>Canon PowerShot S400 4MP Digital Camera w/ 3x Optical Zoom
</ Title>
<UPC>013803023961</UPC>
<Weight Units="pounds">0.41</Weight>
<Width Units="inches">3.43</Width>
</ItemAttributes>
</Item>
</Items>
</ItemLookupResponse>

The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=ItemLookup
&ItemId=B00008OE6I
&ResponseGroup=ItemAttributes

Offers Response Group

Description

The *Offers* response group is a parent response group that returns the contents of the OfferSummary response group plus, by default, all "New" offer listings. For each offer listing, this response groups will return the SellerId and the MerchantId, as well as the offer listing condition, sub-condition, and description. *Offers* response group provides information about product availability and the

product price which are equivalent to *Availability* and *Service Price* in the proposed quality criteria classification in Section 3.3.

REST Sample Response and Request

Sample Response (REST)


```
<?xml version="1.0" encoding="UTF-8"?>
<ItemLookupResponse
xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>1PZJ2MKA8YY3452P0PZX</RequestId>
<Arguments>
<Argument Name="MerchantId" Value="All"/>
<Argument Name="Service" Value="AWSECommerceService"/>
<Argument Name="AssociateTag" Value="[Your Associate ID Here]"/>
<Argument Name="SubscriptionId" Value="[Your Subscription ID Here]"/>
<Argument Name="ItemId" Value="0439358078"/>
<Argument Name="ResponseGroup" Value="Offers"/>
<Argument Name="Operation" Value="ItemLookup"/>
</Arguments>
</OperationRequest>
<Items>
<Request>
<IsValid>True</IsValid>
</Request>
<Item>
<ASIN>0439358078</ASIN>
<OfferSummary>
<LowestNewPrice>
<Amount>514</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$5.14</FormattedPrice>
</LowestNewPrice>
<LowestUsedPrice>
<Amount>525</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$5.25</FormattedPrice>
</LowestUsedPrice>
<LowestCollectiblePrice>
<Amount>957</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$9.57</FormattedPrice>
</LowestCollectiblePrice>
<TotalNew>48</TotalNew>
<TotalUsed>46</TotalUsed>
<TotalCollectible>7</TotalCollectible>
<TotalRefurbished>0</TotalRefurbished>
```

```
</OfferSummary>
<Offers>
<TotalOffers>48</TotalOffers>
<TotalOfferPages>5</TotalOfferPages>
<Offer>
<Seller>
<SellerId>ASYDZOX0HKBSE</SellerId>
</Seller>
<OfferAttributes>
<Condition>New</Condition>
<SubCondition>new</SubCondition>
<ConditionNote>100% Brand New! - Ships Today! Identical to
Copy! *We recommend Expedited Shipping option for much faster mail delivery</
ConditionNote>
</OfferAttributes>
<OfferListing>
<OfferListingId>fGC28xteSrZMVrPT%2BTRkFtuDQaiixLJKXzIWLQqk295vz96a7M4f%2BQi4
z RQlYyi9QAYXPhyqM2aThqdd8YA1aIr3SxsQ7HMB</OfferListingId>
<ExchangeId>Y01Y0538529Y2514641</ExchangeId>
<Price>
<Amount>514</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$5.14</FormattedPrice>
</Price>
<Availability>Usually ships in 1-2 business days</Availability>
</OfferListing>
</Offer>
</Offers>
</Item>
</Items>
</ItemLookupResponse>
```


The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=ItemLookup
&ItemId=0439358078
&MerchantId=All
&ResponseGroup=Offers

Reviews Response Group

Description

The *Reviews* response group provides a list of customer reviews, an average rating (1 to 5 stars) that is equivalent to the product *Reputation* (see Section **Error! Reference source not found.**) in the proposed quality criteria classification, and the total number of reviews for each item in the response. Each customer review will contain the rating, summary, date of review, and full review text.

REST Sample Response and Request

<?xml version="1.0" encoding="UTF-8"?>
<ItemLookupResponse xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>01WJMPWPKSVTA7B567M2</RequestId>
<Arguments>
<Argument Name="Service" Value="AWSECommerceService"/>
<Argument Name="AssociateTag" Value="*[Your Associate ID Here]*"/>
<Argument Name="SubscriptionId" Value="*[Your Subscription ID Here]*"/>
<Argument Name="ItemId" Value="0060006781"/>
<Argument Name="ResponseGroup" Value="Reviews"/>
<Argument Name="Operation" Value="ItemLookup"/>
</Arguments>
</OperationRequest>
<Items>
<Request>
<IsValid>True</IsValid>
</Request>
<Item>
<ASIN>0060006781</ASIN>
<CustomerReviews>
<AverageRating>3.95</AverageRating>
<TotalReviews>20</TotalReviews>
<Review>
<ASIN>0060006781</ASIN>
<Rating>4</Rating>
<HelpfulVotes>9</HelpfulVotes>
<TotalVotes>11</TotalVotes>
<Date>2003-06-13</Date>
<Summary>It's in the genes, just not in the way we thought.</Summary>
</Review>
</CustomerReviews>
</Item>
</Items>
</ItemLookupResponse>

The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=ItemLookup
&IdType=ASIN
&ItemId=0060006781
&ResponseGroup=Reviews

Seller Response Group

Description

The Seller response group provides the seller ID, nickname, average feedback rating which is equivalent to the seller *Reputation* in the proposed quality criteria classification, description, and location for each seller in the response.

REST Sample Response and Request

Sample Response (REST)

```
<?xml version="1.0" encoding="UTF-8"?>
 <SellerLookupResponse xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>0VFWPQA26CMF97MGFKXJ</RequestId>
<Arguments>
<Argument Name="Service" Value="AWSECommerceService"/>
<Argument Name="AssociateTag" Value="[Your Associate ID Here]"/>
<Argument Name="SubscriptionId" Value="[Your Subscription ID Here]"/>
<Argument Name="ResponseGroup" Value="Seller"/>
<Argument Name="Operation" Value="SellerLookup"/>
<Argument Name="SellerId" Value="A3ENSIQ3ZA4FFN"/>
</Arguments>
</OperationRequest>
<Sellers>
<Request>
<IsValid>True</IsValid>
</Request>
<Seller>
<SellerId>A3ENSIQ3ZA4FFN</SellerId>
<SellerName>abebooks.com</SellerName>
<Nickname>abebooks</Nickname>
<Location>
<City>Pt. Roberts</City>
<State>WA</State>
</Location>
<AverageFeedbackRating>4.39</AverageFeedbackRating>
<TotalFeedback>149642</TotalFeedback>
<TotalFeedbackPages>29929</TotalFeedbackPages>
<SellerFeedback>
<Feedback>
<Rating>4</Rating>
<Comment>excellent condition and service if a little lengthy in
overseas delivery time</Comment>
<Date>2004-09-28T05:41+0000</Date>
<RatedBy>A1J4CF92QNWOAE</RatedBy>
</Feedback>
</SellerFeedback>
</Seller>
</Sellers>
```

</SellerLookupResponse>

The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=SellerLookup
&SellerId=A3ENSIQ3ZA4FFN
&FeedbackPage=1
&ResponseGroup=Seller

TransactionDetails Response Group

Description

The *TransactionDetails* response group provides information about Amazon transactions, including the seller ID, the condition of the transaction, the date of the transaction, and the total dollar amount of the transaction which is equivalent to *Execution Price* in the proposed quality criteria classification. *TransactionDetails* does not return information about the items that were purchased or about the customers who completed the transaction.

REST Sample Response and Request

Sample Response (REST)

<?xml version="1.0" encoding="UTF-8"?>
<TransactionLookupResponse xmlns="http://webservices.amazon.com/AWSECommerceService/
2004-08-01">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1)"/>
</HTTPHeaders>
<RequestId>0JGK2H9TFCDSXN0BTY6B</RequestId>
<Arguments>
<Argument Name="Service" Value="AWSECommerceService"/>
<Argument Name="AssociateTag" Value="*[Your Associate ID Here]*"/>
<Argument Name="SubscriptionId" Value="*[Your Subscription ID Here]*"/>
<Argument Name="TransactionId" Value="104-1867480-8536729"/>
<Argument Name="ResponseGroup" Value="TransactionDetails"/>
<Argument Name="Operation" Value="TransactionLookup"/>
</Arguments>
</OperationRequest>
<Transactions>
<Request>
<IsValid>True</IsValid>
</Request>
<Transaction>

<TransactionId>104-1867480-8536729</TransactionId>
<SellerId>ATVPDKIKX0DER</SellerId>
<Condition>Complete</Condition>
<TransactionDate>2004-06-14T21:51:53</TransactionDate>
<TransactionDateEpoch>1087249913</TransactionDateEpoch>
<SellerName>Amazon.com</SellerName>
<Tax>
<Amount>2.49</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$2.00</FormattedPrice>
</Tax>
<ShippingCharge>
<Amount>4.98</Amount>
<CurrencyCode>USD</CurrencyCode>
<FormattedPrice>$4.00</FormattedPrice>
</ShippingCharge>
<Shipments>
<Shipment>
<Condition>Shipped</Condition>
<DeliveryMethod>Mail</DeliveryMethod>
<ShipmentItems>
<TransactionItemId>jjsnptouplox</TransactionItemId>
<TransactionItemId>jjsnptouorox</TransactionItemId>
</ShipmentItems>
</Shipment>
</Shipments>
</Transaction>
</Transactions>
</TransactionLookupResponse>

The Request that Generated the Response (REST)

http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&SubscriptionId=*[Your Subscription ID Here]*
&AssociateTag=*[Your Associate ID Here]*
&Operation=TransactionLookup
&TransactionId=104-1867480-8536729
&ResponseGroup=TransactionDetails

Appendix E: Using SOAP Request to Access Amazon E-Commerce Service

A simple ASP.NET Web application is taken from [146] to use SOAP request to access ECS. In order to access Amazon E-Commerce Service server, it is required to add a Web reference to Amazon Web Services by selecting the Project | Add Web Reference menu option from Visual Studio .NET and then enter the following URL in the address text box as shown in Figure E- 1 Add Web Reference:

http://soap.amazon.com/schemas2/AmazonWebServices.wsdl

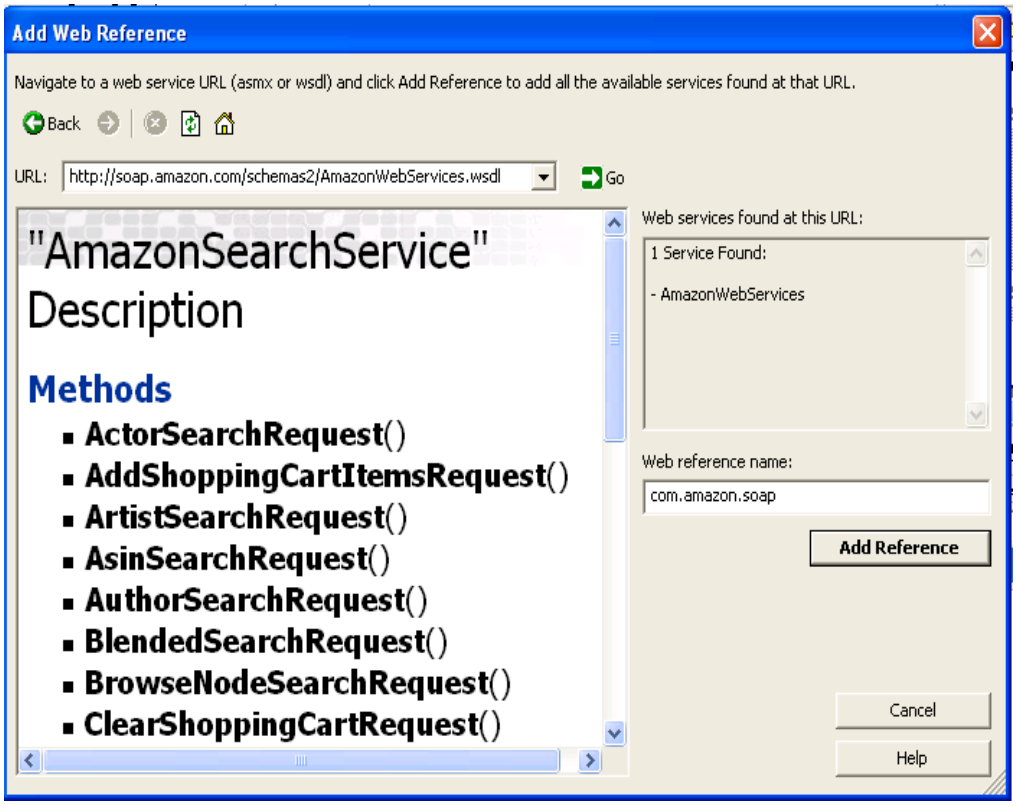

Figure E- 1 Add Web Reference

The source code of ASP.NET Web application taken from [146]:

```
using System;
using System.Threading;
using System.Drawing;
```

Appendix E: Using SOAP Request to Access Amazon E-Commerce Service

```
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using amazonCache.com.amazon.webservices;
namespace amazonCache
{
public class Form1 : System.Windows.Forms.Form
{
 private System.Windows.Forms.Button button1;
 private System.Windows.Forms.TextBox textBox1;
 private System.Windows.Forms.Label label1;
 static void Main()
 {
  Application.Run(new Form1());
 }
 private void button1_Click(object sender, System.EventArgs e)
 {
  ItemSearchResponse response;
   AWSECommerceService aws=new AWSECommerceService();
   ItemSearchRequest request=new ItemSearchRequest();
   request.SearchIndex="Books";
   request.Power="title:"+textBox1.Text;
   request.ResponseGroup=new string[] {"Large"};
   request.Sort="salesrank";

   ItemSearchRequest[] requests=new
                ItemSearchRequest[] {request};

   ItemSearch itemSearch =new ItemSearch();
   itemSearch.AssociateTag="webservice1-20";
   itemSearch.SubscriptionId=" 1NC71HN9R7AE4KJ1G3G2";
   itemSearch.Request=requests;
   response=aws.ItemSearch(itemSearch);
   Items info =response.Items[0];
   Item[] items=info.Item;
   label1.Text="";
   for(int i=0; i<items.Length;i++)
   {
   Item item=items[i];
   label1.Text+="Book
                Title:"+item.ItemAttributes.Title+"<br/>";
   }
  }
 }
```

Appendix F: REST Request and XML Data Result

By typing the REST request1 as shown in Figure F- 1 in the address bar in the Internet explorer and hit "Go" button, the following XML result is displayed:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService &
SubscriptionId=1NC71HN9R7AE4KJ1G3G2 &Operation=ItemSearch &Title=web
services & SearchIndex=Books &MerchantId=All &ResponseGroup=Item Attributes.
OfferFull
```

Figure F- 1REST Request 1

```
<?xml version="1.0" encoding="UTF-8" ?>
  <ItemSearchResponse
  xmlns="http://webservices.amazon.com/AWSECommerceService/2005-10-05">
  <OperationRequest>
  <HTTPHeaders>
  <Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
  SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50215)" />
  </HTTPHeaders>
  <RequestId>0VX4GE3BHXPQDZ60AFQA</RequestId>
  <Arguments>
  <Argument Name="MerchantId" Value="All" />
  <Argument Name="Service" Value="AWSECommerceService" />
  <Argument Name="Title" Value="web services" />
  <Argument Name="SearchIndex" Value="Books" />
  <Argument Name="SubscriptionId" Value="1NC71HN9R7AE4KJ1G3G2" />
  <Argument Name="ResponseGroup" Value="ItemAttributes,OfferFull" />
  <Argument Name="Operation" Value="ItemSearch" />
  </Arguments>
  <RequestProcessingTime>0.901411056518555</RequestProcessingTime>
  </OperationRequest>
  <Items>
  <Request>
  <IsValid>True</IsValid>
  <ItemSearchRequest>
  <MerchantId>All</MerchantId>
  <ResponseGroup>OfferFull</ResponseGroup>
  <ResponseGroup>ItemAttributes</ResponseGroup>
  <SearchIndex>Books</SearchIndex>
  <Title>web services</Title>
  </ItemSearchRequest>
  </Request>
  <TotalResults>933</TotalResults>
  <TotalPages>94</TotalPages>
  <Item>
    <ASIN>0131428985</ASIN>
```

Appendix F: REST Request and XML Data Result

<ItemAttributes>
  <Author>**Thomas Erl**</Author>
  <ISBN>**0131428985**</ISBN>
  <Height Units="**hundredths-inches**">**145**</Height>
  <Length Units="**hundredths-inches**">**918**</Length>
  <Weight Units="**hundredths-pounds**">**231**</Weight>
  <ProductGroup>**Book**</ProductGroup>
  <PublicationDate>**2004-04-16**</PublicationDate>
  <Publisher>**Prentice Hall PTR**</Publisher>
  <Title>**Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services**</Title>
</ItemAttributes>
<Offers>
  <TotalOffers>**35**</TotalOffers>
  <TotalOfferPages>**4**</TotalOfferPages>
<Offer>
<Seller>
  <SellerId>**A2ZGNN73WLXVFQ**</SellerId>
  <Nickname>**a1books**</Nickname>
  <AverageFeedbackRating>**4.5**</AverageFeedbackRating>
</Seller>
<Price>
  <Amount>**2243**</Amount>
  <CurrencyCode>**USD**</CurrencyCode>
  <FormattedPrice>**$28.21**</FormattedPrice>
</Price>
<Availability>**Usually ships in 1-2 business days**</Availability>
  </Offer>
<Offer>
<Seller>
  <SellerId>**A2PH0OU9DK0NPM**</SellerId>
  <Nickname>**fantastic_shopping**</Nickname>
  <AverageFeedbackRating>**4.5**</AverageFeedbackRating>
</Seller>
<Price>
  <Amount>**2243**</Amount>
  <CurrencyCode>**USD**</CurrencyCode>
  <FormattedPrice>**$24.14**</FormattedPrice>
</Price>
<Availability>**only 70% left in stock**</Availability>
</Offer>
…….
</Offers>
</Item>
<Item>
<ASIN>**0131488740**</ASIN>
<ItemAttributes>
<Author>**Sanjiva Weerawarana**</Author>
<Author>**Francisco Curbera**</Author>
<Author>**Frank Leymann**</Author>
<Author>**Tony Storey**</Author>
<Author>**Donald F. Ferguson**</Author>
<ISBN>**0131488740**</ISBN>

<Height Units="**hundredths-inches**">**82**</Height>
<Length Units="**hundredths-inches**">**938**</Length>
<Weight Units="**hundredths-pounds**">**136**</Weight>
<ProductGroup>**Book**</ProductGroup>
<PublicationDate>**2005-03-22**</PublicationDate>
<Publisher>**Prentice Hall PTR**</Publisher>
<Title>**Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More**</Title>
<Offers>
<TotalOffers>**47**</TotalOffers>
<TotalOfferPages>**5**</TotalOfferPages>
<Offer>
<Seller>
<SellerId>**A3B9364CV8QDO9**</SellerId>
<Nickname>**amz_book**</Nickname>
<AverageFeedbackRating>**3**</AverageFeedbackRating>
</Seller>
<Price>
<Amount>**2895**</Amount>
<CurrencyCode>**USD**</CurrencyCode>
<FormattedPrice>**$29.95**</FormattedPrice>
</Price>
<Availability>**Limited availability**</Availability>
<Offer>
<Seller>
<SellerId>**A1KIF2Y9A1PQYE**</SellerId>
<Nickname>**allnewbooks**</Nickname>
<AverageFeedbackRating>**2.6**</AverageFeedbackRating>
</Seller>
<Price>
<Amount>**2924**</Amount>
<CurrencyCode>**USD**</CurrencyCode>
<FormattedPrice>**$32.34**</FormattedPrice>
</Price>
<Availability>S**pecial order**</Availability>
</Offer>
</Item>
<Item>
<ASIN>**0321369440**</ASIN>
<ItemAttributes>
<Author>**Mike Andrews**</Author>
<Author>**James A. Whittaker**</Author>
<ISBN>**0321369440**</ISBN>
<Height Units="**hundredths-inches**">**65**</Height>
<Length Units="**hundredths-inches**">**916**</Length>
<Weight Units="**hundredths-pounds**">**108**</Weight>
<ProductGroup>**Book**</ProductGroup>
<PublicationDate>**2006-02-02**</PublicationDate>
<Publisher>**Addison-Wesley Professional**</Publisher>
<Title>**How to Break Web Software: Functional and Security Testing of Web Applications and Web Services**</Title>
</ItemAttributes>
<Offer>

<Merchant>
<MerchantId>**ATVPDKIKX0DER**</MerchantId>
<Name>**Amazon.com**</Name>
</Merchant>
<Price>
<Amount>**2204**</Amount>
<CurrencyCode>**USD**</CurrencyCode>
<FormattedPrice>**$22.04**</FormattedPrice>
</Price>
<Availability>Usually ships in 10 to 14 days</Availability>
</Offer>
<Offer>
<Seller>
<SellerId>**A2E9OWRCF7T08Y**</SellerId>
<Nickname>**pbshopus**</Nickname>
<AverageFeedbackRating>**4**</AverageFeedbackRating>
</Seller>
<Price>
<Amount>**2329**</Amount>
<CurrencyCode>**USD**</CurrencyCode>
<FormattedPrice>**$43.54**</FormattedPrice>
</Price>
<Availability>Usually ships in 1-2 business days</Availability>
</Offer>
……..
</Item>
<Item>
<ASIN>**0131463071**</ASIN>
<ItemAttributes>
<Author>**Christopher Steel**</Author>
<Author>**Ramesh Nagappan**</Author>
<Author>**Ray Lai**</Author>
<ISBN>**0131463071**</ISBN>
<Height Units="**hundredths-inches**">**220**</Height>
<Length Units="**hundredths-inches**">**938**</Length>
<Weight Units="**hundredths-pounds**">**377**</Weight>
<ProductGroup>**Book**</ProductGroup>
<PublicationDate>**2005-10-14**</PublicationDate>
<Publisher>**Prentice Hall PTR**</Publisher>
<Title>**Core Security Patterns: Best Practices and Strategies for J2EE(TM), Web Services, and Identity Management (Core)** </Title>
</ItemAttributes>
<Offer>
<Seller>
<SellerId>**AT7MC65GYVR0L**</SellerId>
<Nickname>**backalleytextbooks**</Nickname>
<AverageFeedbackRating>**2**</AverageFeedbackRating>
</Seller>
<Price>
<Amount>**3556**</Amount>
<CurrencyCode>**USD**</CurrencyCode>
<FormattedPrice>**$36.41**</FormattedPrice>

```
</Price>
  <Availability> Limited availability </Availability>
</Offer>
<Offer>
<Seller>
  <SellerId>A1MD3EN9VM2K1F</SellerId>
  <Nickname>fun-for-all58</Nickname>
  <AverageFeedbackRating>3.5</AverageFeedbackRating>
</Seller>
<Price>
  <Amount>3585</Amount>
  <CurrencyCode>USD</CurrencyCode>
  <FormattedPrice>$33.85</FormattedPrice>
</Price>
  <Availability>Usually ships in 1-2 business days</Availability>
</Offer>
……
</Item>
<Item>
  <ASIN>0321180860</ASIN>
<ItemAttributes>
  <Author>Eric Newcomer</Author>
  <Author>Greg Lomow</Author>
  <ISBN>0321180860</ISBN>
  <Height Units="hundredths-inches">88</Height>
  <Length Units="hundredths-inches">920</Length>
  <Weight Units="hundredths-pounds">156</Weight>
  <ProductGroup>Book</ProductGroup>
  <PublicationDate>2004-12-14</PublicationDate>
  <Publisher>Addison-Wesley Professional</Publisher>
  <Title>Understanding SOA with Web Services (Independent Technology Guides)</Title>
</ItemAttributes>
<Offer>
<Seller>
  <SellerId>AHNEEZ9CVAP3Q</SellerId>
  <Nickname>superbookdeals</Nickname>
  <AverageFeedbackRating>4.6</AverageFeedbackRating>
</Seller>
<Price>
  <Amount>2463</Amount>
  <CurrencyCode>USD</CurrencyCode>
  <FormattedPrice>$24.63</FormattedPrice>
</Price>
  <Availability>Usually ships in 1-2 business days</Availability>
</Offer>
<Offer>
<Seller>
  <SellerId>A2ZGNN73WLXVFQ</SellerId>
  <Nickname>a1books</Nickname>
  <AverageFeedbackRating>4.5</AverageFeedbackRating>
</Seller>
<Price>
```

```
   <Amount>2505</Amount>
   <CurrencyCode>USD</CurrencyCode>
   <FormattedPrice>$25.05</FormattedPrice>
</Price>
   <Availability>Usually ships in 1-2 business days</Availability>
</Offer>
……..
</Item>
……
</Items>
</ItemSearchResponse>
```

By typing the REST request2 as shown in Figure F- 1 in the address bar in the Internet explorer and hit "Go" button, the following XML result is displayed:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService &
SubscriptionId=1NC71HN9R7AE4KJ1G3G2&AssociateTag=webservice1-20
&Operation=SellerLookup &SellerId=A3E0GMZ4YFS6AQ & ResponseGroup=Seller
```

Figure F- 2 REST Request 2

```
<?xml version="1.0" encoding="UTF-8" ?>
<SellerLookupResponse
xmlns="http://webservices.amazon.com/AWSECommerceService/2005-10-05">
<OperationRequest>
<HTTPHeaders>
<Header Name="UserAgent" Value="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50215)" />
</HTTPHeaders>
<RequestId>1P846BC2TQGNJ4GRQS69</RequestId>
<Arguments>
<Argument Name="AssociateTag" Value="webservice1-20" />
<Argument Name="SubscriptionId" Value="1NC71HN9R7AE4KJ1G3G2" />
<Argument Name="ResponseGroup" Value="Seller" />
<Argument Name="Operation" Value="SellerLookup" />
<Argument Name="Service" Value="AWSECommerceService" />
<Argument Name="SellerId" Value="A2PH0OU9DK0NPM" />
</Arguments>
<RequestProcessingTime>0.149191856384277</RequestProcessingTime>
</OperationRequest>
<Sellers>
<Request>
<IsValid>True</IsValid>
<SellerLookupRequest>
<ResponseGroup>Seller</ResponseGroup>
<SellerId>A2PH0OU9DK0NPM</SellerId>
</SellerLookupRequest>
</Request>
<Seller>
<SellerId>A2PH0OU9DK0NPM</SellerId>
```

<Nickname>**fantastic_shopping**</Nickname>
<GlancePage>**http://www.amazon.com/gp/help/seller/at-a-glance.html?seller=A2PH0OU9DK0NPM&marketplaceSeller=1**</GlancePage>
<Location>
<City>**Olsmar**</City>
<State>**FL**</State>
</Location>
<AverageFeedbackRating>**4.5**</AverageFeedbackRating>
<TotalFeedback>**77105**</TotalFeedback>
<TotalFeedbackPages>**15421**</TotalFeedbackPages>
<SellerFeedback>
<Feedback>
<Rating>**5**</Rating>
<Comment>**Perfect condition, fast and easy...they were great to work with and would do it again!**</Comment>
<Date>**2006-08-01T09:36+0000**</Date>
<RatedBy>**AFB4TV461N47C**</RatedBy>
</Feedback>
<Feedback>
<Rating>**4**</Rating>
<Comment>**book is in great codition, would be better if seller had contacted me about the delay. otherwise a good, responsible seller**</Comment>
<Date>**2006-08-01T07:46+0000**</Date>
<RatedBy>**A2VGXBM60L3X**</RatedBy>
</Feedback>
<Feedback>
<Rating>**5**</Rating>
<Comment>**Thank you very much! A great book, quick shipping**</Comment>
<Date>**2006-08-01T07:18+0000**</Date>
<RatedBy>**AQKF5TAQWT08E**</RatedBy>
</Feedback>
<Feedback>
<Rating>**5**</Rating>
<Comment>**Perfect condition and arrived before delivery estimate**</Comment>
<Date>**2006-08-01T06:39+0000**</Date>
<RatedBy>**A2JKT9E6JK093T**</RatedBy>
</Feedback>
<Feedback>
<Rating>**5**</Rating>
<Comment>**The item was exactly as described, well-packed and arrived promptly. Great.**</Comment>
<Date>**2006-08-01T05:46+0000**</Date>
<RatedBy>**A1PA3JWMNLQ913**</RatedBy>
</Feedback>
…….
</SellerFeedback>
</Seller>
</Sellers>
</SellerLookupResponse>

Appendix G: Amazon E-Commerce (ECS) database

Table G- 1 Amazon Database

Product Name	Seller Name	Availability	Price	Seller Reputation	Seller URL
Service-Oriented Architecture	hebertbooks	99	24.1	3.4	http://www.amazon.com/seller=A1RAFT0AR298LX
	fantastic_shopping	87	24.14	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	fun-for-all58	75	24.3	3.5	http://www.amazon.com/seller=A1MD3EN9VM2K1F
	yaleiz	80	27.99	4	http://www.amazon.com/seller=A1MOV0BA9DKUFU
	a1books	97	28.21	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	Amazon.com	99	28.34	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	allnewbooks	80	29.19	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	95	30.07	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
Web Services Platform Architecture	fantastic_shopping	90	29.94	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	amz_book	78	29.95	3	http://www.amazon.com/seller=A3B9364CV8QDO9
	a1books	98	31.05	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	Amazon.com	99	31.49	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	allnewbooks	65	32.34	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	30	33.41	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	thebookrackrh	75	34.09	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	a1books_nj	95	34.11	3.6	http://www.amazon.com/seller=A3E0GMZ4YFS6AQ
	alphacraze	82	34.34	3.4	http://www.amazon.com/seller=A2NT0F3A6LH7YD
	alphacrazeoutlet	97	34.34	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1
J2EE Web Services	bookbensara	95	29.75	4.2	http://www.amazon.com/seller=A3H8H6KI3KCVA5
	fantastic_shopping	85	34.63	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	Amazon.com	99	34.64	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	a1books	98	35.04	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	thebookrackrh	80	35.4	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	allnewbooks	95	35.49	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	20	37.2	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	alphacrazeoutlet	79	37.93	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1
	alphacraze	99	37.93	3.4	http://www.amazon.com/seller=A2NT0F3A6LH7YD
	a1books_nj	96	38.51	3.6	http://www.amazon.com/seller=A3E0GMZ4YFS6AQ
How to Break Web Software	tudent2studentbook	82	20.95	2.5	http://www.amazon.com/seller=A1BU1B4BZ1L0UY
	Amazon.com	95	22.04	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	indoobestsellers	69	25.67	3.5	http://www.amazon.com/seller=A1C3QU77DDT2KW
	powells_books	90	34.99	2.8	http://www.amazon.com/seller=AZPQKLIWQKVZ
	pbshop	37	41.68	4.3	http://www.amazon.com/seller=AGLPMRINU0Q3T
	pbshopus	96	43.54	4	http://www.amazon.com/seller=A2E9OWRCF7T08Y
	the_book_depositor	80	43.68	4	http://www.amazon.com/seller=A3TJVJMBQL014A
	bestdictionaries	99	60.08	4.7	http://www.amazon.com/seller=AXQ97OWZ5BK0
Core Security Patterns	fun-for-all58	98	33.85	3.5	http://www.amazon.com/seller=A1MD3EN9VM2K1F
	bargainbookswest	20	34.5	2.9	http://www.amazon.com/seller=A1XZPX0I00ZMJB
	fantastic_shopping	99	36.4	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	Amazon.com	99	37.79	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	a1books	65	38.15	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	allnewbooks	85	38.64	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	thebookrackrh	97	39.2	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	amz_book	84	39.95	3	http://www.amazon.com/seller=A3B9364CV8QDO9
	caiman_com	96	40.59	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6

Appendix G: Amazon E-Commerce (ECS) database

Professional PHP Web Services	ultimatediscountbo	99	5.93	2	http://www.amazon.com/seller=A1P7V4VA92G0N5
	westcoast_books	90	5.93	3.2	http://www.amazon.com/seller=A21YEUH7S5G16G
	hbytes	98	22.99	2.5	http://www.amazon.com/seller=AG28AH8GM6N4A
	torianme	75	24.5	3.5	http://www.amazon.com/seller=AAHR384CN72UL
	smartlion	98	49.99	4	http://www.amazon.com/seller=A31F8XEATOE7XI
Business Process Execution Language for Web Services	Amazon.com	99	69.99	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	pbshopus	84	73.97	4.3	http://www.amazon.com/seller=A2E9OWRCF7T08Y
	caiman_com	60	82.35	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	bigrockmedia_dot	98	83.19	3	http://www.amazon.com/seller=A1LZ6NN9EPDRKV
	mediacrazy_com	25	83.36	2.5	http://www.amazon.com/seller=A1ZGIGWL4Q5LD0
	bigrockmedia_com	95	83.49	2.8	http://www.amazon.com/seller=AQUOVJUDTTXEN
	movieweb_com	87	83.65	3.6	http://www.amazon.com/seller=A3FXKQSDE6Q9HK
	oddbanana_com	97	84.01	4	http://www.amazon.com/seller=A1L1LPVB9RINQ5
Building Web Services with Java	fantastic_shopping	95	31.48	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	Amazon.com	99	31.49	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	a1books	78	32.04	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	allnewbooks	82	32.34	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	98	34.06	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	alphacraze	97	34.34	3.4	http://www.amazon.com/seller=A2NT0F3A6LH7YD
	alphacrazeoutlet	66	34.34	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1
	pbshopus	98	34.44	4	http://www.amazon.com/seller=A2E9OWRCF7T08Y
Understanding SOA with Web Services	fantastic_shopping	80	24.81	4.5	http://www.amazon.com/seller=A2PH0OU9DK0NPM
	superbookdeals	96	25.04	2.5	http://www.amazon.com/seller=AHNEEZ9CVAP3Q
	a1books	82	25.05	4.5	http://www.amazon.com/seller=A2ZGNN73WLXVFQ
	Amazon.com	99	25.19	5	http://www.amazon.com/seller=ATVPDKIKX0DER
	thebookrackrh	39	25.71	2.8	http://www.amazon.com/seller=A1SSUO20DOKMFO
	amz_book	95	25.95	3	http://www.amazon.com/seller=A3B9364CV8QDO9
	allnewbooks	68	26.04	2.6	http://www.amazon.com/seller=A1KIF2Y9A1PQYE
	caiman_com	98	26.73	4.4	http://www.amazon.com/seller=A1MSHKP33DCC6
	a1books_nj	86	27.28	3.6	http://www.amazon.com/seller=A3E0GMZ4YFS6AQ
	lphacrazeoutlet	98	27.39	3.7	http://www.amazon.com/seller=A13MCS24BSAIL1

Appendix H: Visual Studio .NET

H.1 Windows Applications and C#

Visual Studio .NET is a tool that Microsoft has created for helping developers to build next generation of application for the .NET platform [24]. Visual Studio .NET is Microsoft's Integrated Development Environment (IDE)- software used to create, run and debug programs [6].

Visual Studio .NET IDE provides a sophisticated environment for visual programming by which the pre-packaged components can be dragged and dropped into an application. Visual Studio .NET's tools facilitate code reuse by making it easy to build applications from existing code [6].

H.1.1Creating Windows Application

Figure H-0-1 displays a Windows application in Visual Studio .NET with project name quality service selection system (QSSS). QSSS is a system used to implement the quality matchmaking process (QMP) to select the best available Web service based on requester's quality preferences and mathematical model. QSSS system displays a graphical user interface (GUI) and contains at least one window. Windows applications execute within the Windows operating system.

The large gray box is called *form* and represents a Windows application. In Figure H-0-1, the form name is "CriteriaSelection.cs" and the programming language is C#. Programmers customize forms by adding *controls* from the *Toolbox.* The *Toolbox* contains reusable software components (or controls) that developers can use them to customize applications. The *form* and *controls* comprise the program's graphical user interface (GUI) [6]. The *Properties* window allows programmers to manipulate form or control properties.

AAppendix G: Amazon E-Commerce (ECS) database
Windows Applications and C#

In quality service selection system (QSSS), many forms are added by selecting (File, Add New Item) then select *Windows Form* as shown in Figure H-0-2. In QSSS, there are five forms: *Criteria Selection*, *Preference Selection, Sub-Criteria Selection, Sub-Preference Selection* and *Requirements Value.* The functions of each form will be explained in the coming section.

From Figure H-0-2, class can be added in QSSS and it is called Utilities. Utilities class contains Matrix class and methods, which is described in Section **Error! Reference source not found.**.

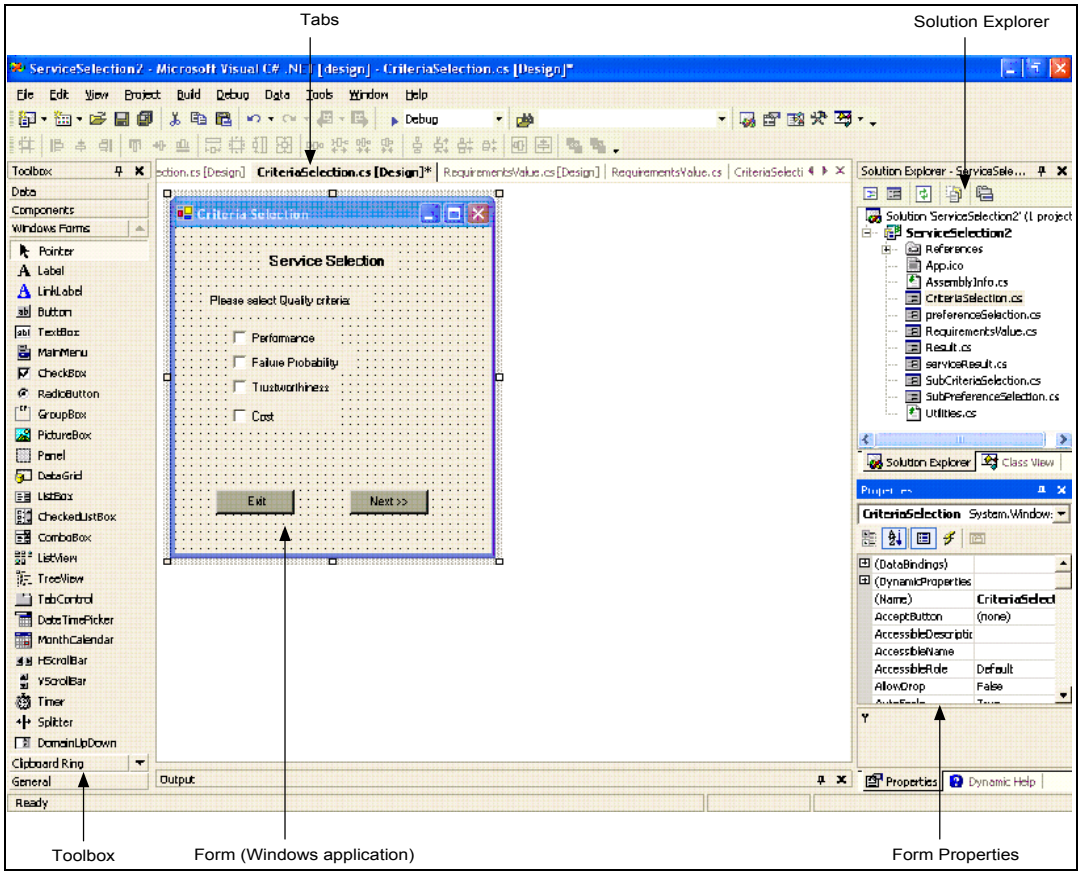

Figure H-0-1 Designing a Windows Application in the Visual Studio .NET IDE

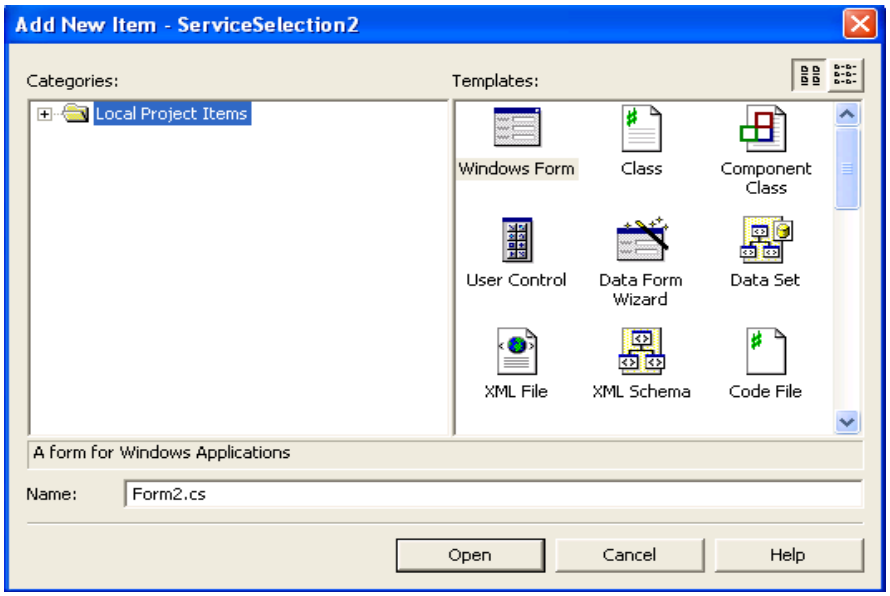

Figure H-0-2 Adding a new Form to a Windows Application

The Framework Class Library (FCL) is used to display the above five forms and these controls (Checkbox, Button, Label, ComboBox, GroupBox, etc.,) in them. A new form, for example *Criteria Selection* form, is created by deriving the main class *Criteria Selection* from the *System.Windows.Forms.Form* class and adding labels(class:System.Windows.Forms.Label),buttons(class:System.Windows.Forms.Button) and checkbox (class:System.Windows.Forms.ChechBox) to the form as shown in **Error! Reference source not found.**.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.OleDb;
namespace ServiceSelection2
{
  public class CriteriaSelection : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        public System.Windows.Forms.CheckBox checkBox1;
        public System.Windows.Forms.CheckBox checkBox2;
        public System.Windows.Forms.CheckBox checkBox3;
        public System.Windows.Forms.CheckBox checkBox4;
```

The FCL is made up of a hierarchy of namespaces that expose classes, structures, interfaces, enumerations, and delegates that can access to these resources. There are more than 20,000 classes in the FCL, all logically grouped in a hierarchical manner. To use an FCL class in the application, it needs to use the `using` statement in C#. `System` is the namespace used for most FCL classes.

The namespaces are logically defined by functionality. For example, the System.Data namespace contains all the functionality available to accessing databases. This namespace is further broken down into `System.Data.OleDb`, which exposes specific functionality for accessing OLEDB data sources.

8.2.1 Visual C# .NET

C# is an object-oriented programming language designed for building a wide range of applications that run on the .NET Framework; it was announced in July 2000 by Anders Hejlsberg and Scott Wiltamuth. C# classes are very similar to C++ classes but there are many differences between C++ and C# as in the following:

- C# does not use header files as C++ does.
- C# supports an XML style of documentation comments marked with ///.
- C# de-emphasizes pointers by inventing *delegates*, which acts like function pointers.
- C# implements structs as a lightweight type very different from classes, whereas structs and classes are very close in C++.
- C# entry point is Main ( ), not main ( ).
- Conditional statements such as if are restricted to Boolean operands in C#.

Source code written in C# is compiled into an Intermediate language (IL) that stores in an executable file called an assembly with an extension of .exe or .dll. An assembly provides information on the assembly's types, version and security requirements.

When the C# program is executed, the assembly is loaded into the Common Language Runtime (CLR). If the security requirements are met, the CLR performs Just in Time (JIT) compilation to convert the Intermediate language (IL) code into native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling, and resource management. The following diagram illustrates the compile-time and run time relationships of C# source code files, the base class libraries, assemblies, and the CLR.

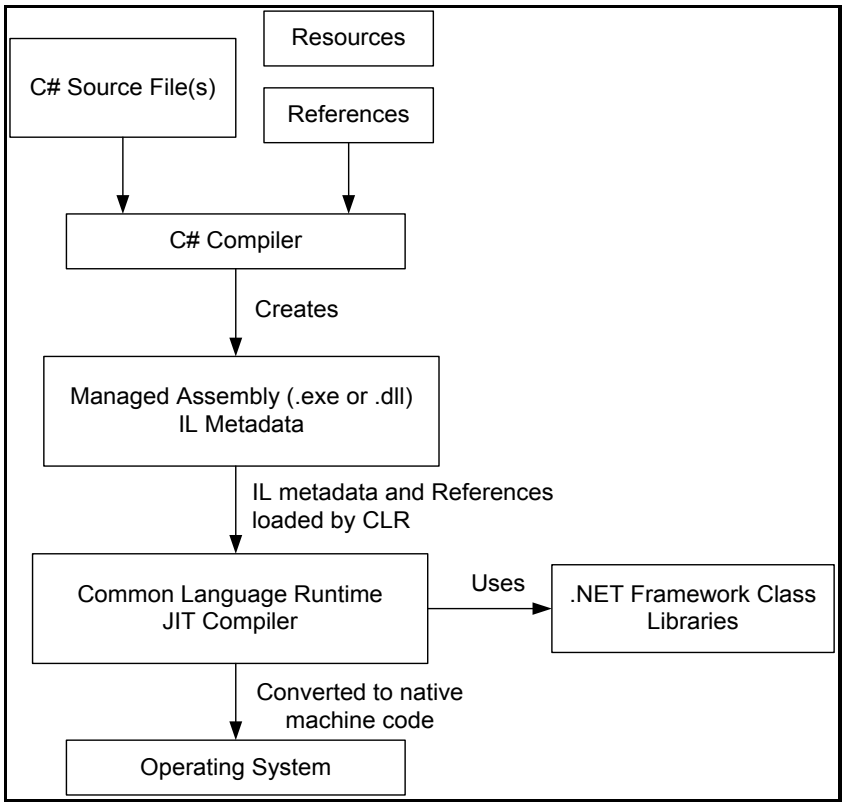

Figure 0H-0-3 Compile time and Run time of C# source code [Taken from [151]]

Language interoperability is a key feature of the .NET Framework. Because the IL code produced by the C# compiler conforms to the Common Type Specification (CTS), IL code generated from C# can interact with code that was generated from the .NET versions of Visual Basic, Visual C++ or Visual J#. A single assembly may contain multiple modules written in different .NET languages, and the types can reference each other as if they were written in the same language [151].

Visual Studio supports Visual C# with a full-featured Code Editor, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET Framework class library provides access to a wide range of operating system services and other useful, well-designed classes that speed up the development cycle significantly.